IBM Resilient

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
   ```
2. 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
   ```
2. 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.

3. 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.

4. 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.

5. Configure Resilient Circuits to run continuously. The following is an example for Red Hat.
   a. Create the unit file as follows:
      ```
      sudo vi /etc/systemd/system/resilient_circuits.service
      ```
   b. 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
      ```
   c. Verify the service unit file permissions using the following command:
      ```
      sudo chmod 664 /etc/systemd/system/resilient_circuits.service
      ```
   d. 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

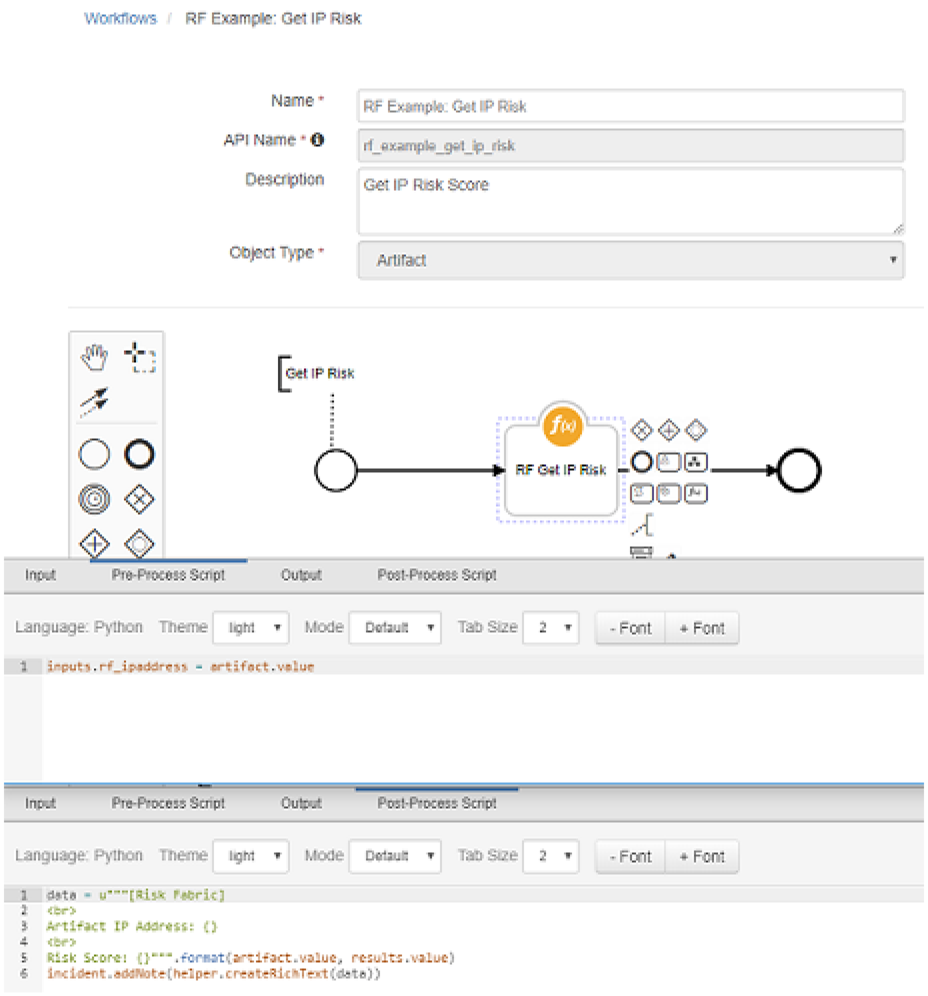
Integration Functions

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	• rf_riskmodelinstanceid: Identifier for the risk model instance being classified	None

Function	Description	Inputs	Outputs
	Acceptable, Investigate, and Violation.	• 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	
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.



Name * RF Example: Get IP Risk

API Name * rf_example_get_ip_risk

Description Get IP Risk Score

Object Type * Artifact

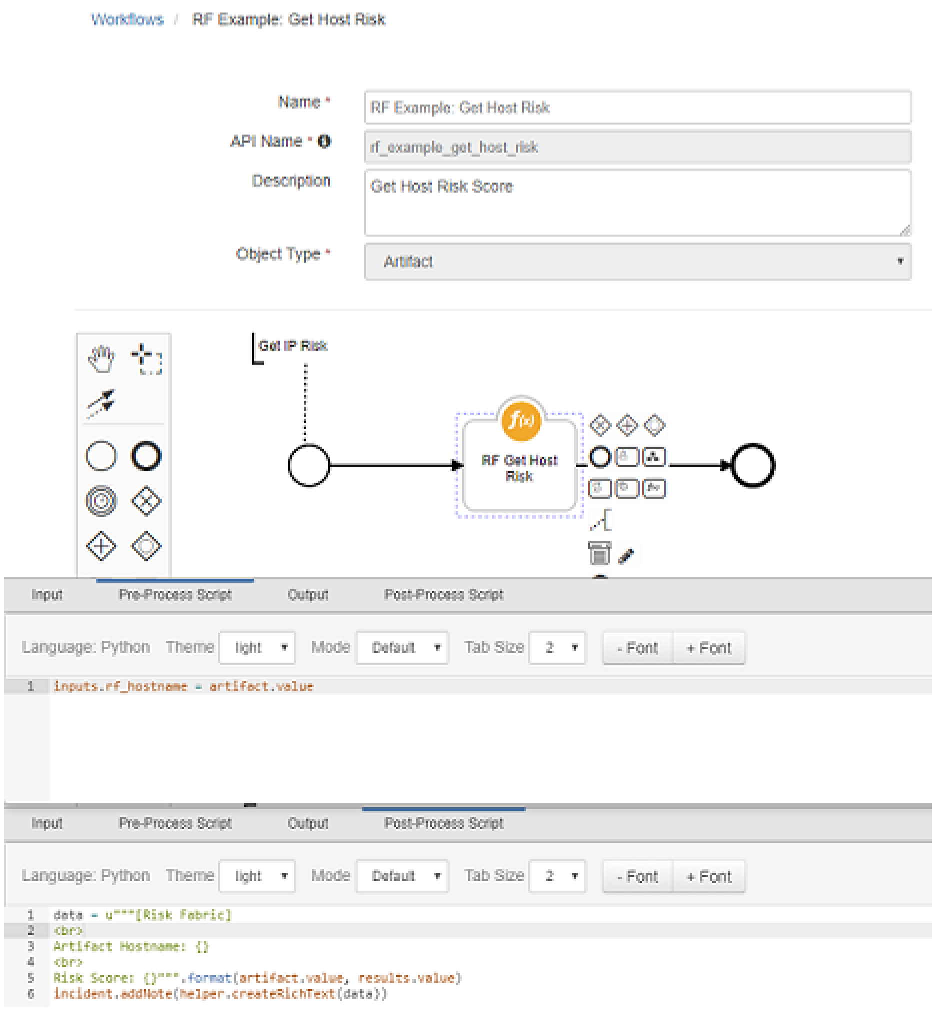
Language: Python Theme light Mode Default Tab Size 2 - Font + Font

```
inputs.rf_ipaddress = artifact.value
```

Language: Python Theme light Mode Default Tab Size 2 - Font + Font

```
data = u"""[Risk Fabric]
<br>
Artifact IP Address: {}
<br>
Risk Score: {}""".format(artifact.value, results.value)
incident.addNote(helper.createRichText(data))
```

- 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.


Workflows / RF Example: Get Host Risk

Name *	RF Example: Get Host Risk
API Name *	rf_example_get_host_risk
Description	Get Host Risk Score
Object Type *	Artifact

Input | Pre-Process Script | Output | Post-Process Script

Language: Python Theme light Mode Default Tab Size 2 - Font + Font

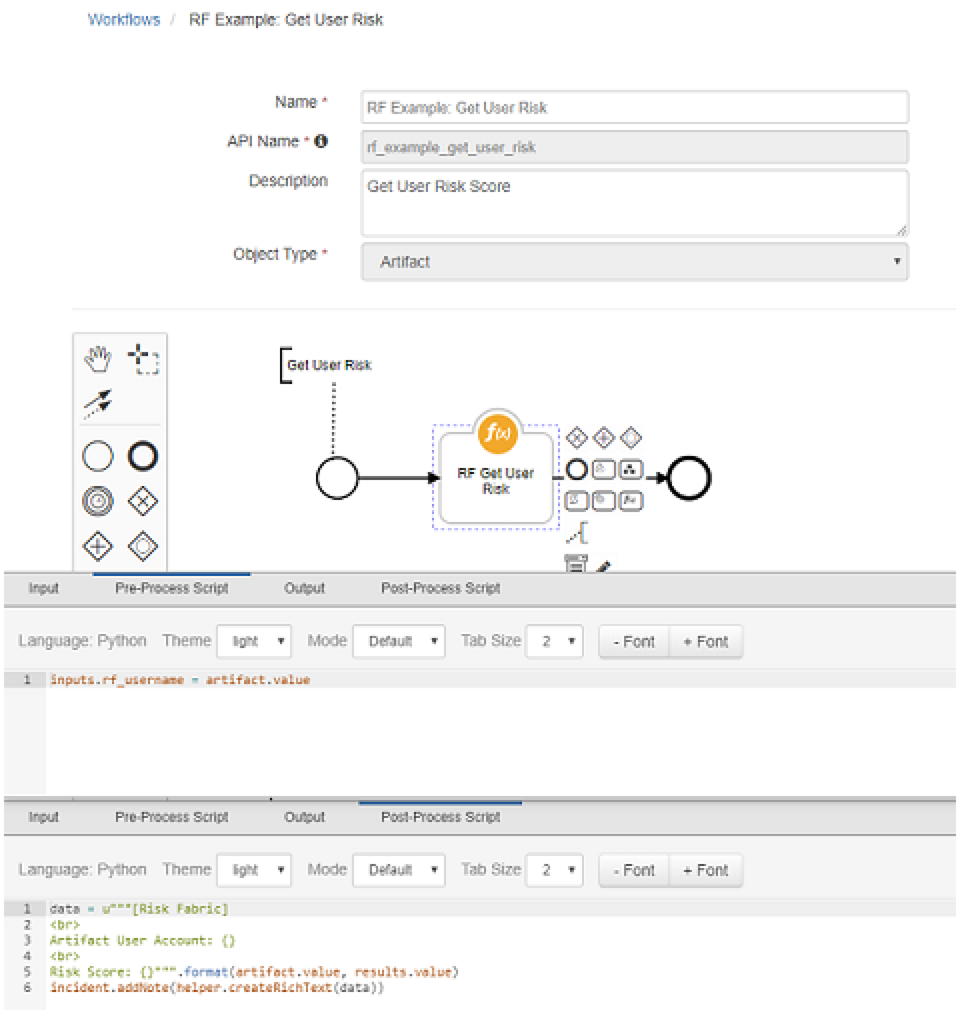
```
inputs.rf_hostname = artifact.value
```

Input | Pre-Process Script | Output | Post-Process Script

Language: Python Theme light Mode Default Tab Size 2 - Font + Font

```
data = u"""[Risk Fabric]
<br>
Artifact Hostname: {}
<br>
Risk Score: {}""".format(artifact.value, results.value)
incident.addNote(helper.createRichText(data))
```

- 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.



Name *: RF Example: Get User Risk
API Name *: rf_example_get_user_risk
Description: Get User Risk Score
Object Type *: Artifact

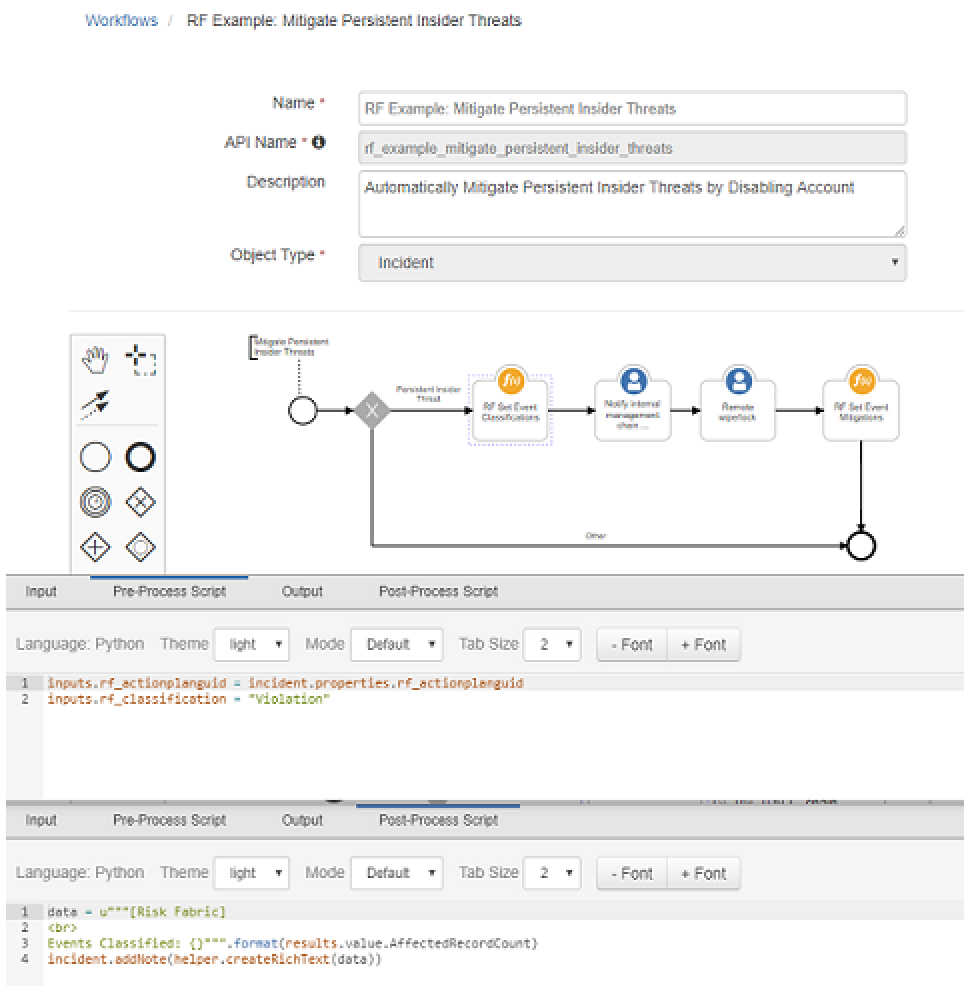
Pre-Process Script — Language: Python, Theme: light, Mode: Default, Tab Size: 2

```
inputs.rf_username = artifact.value
```

Post-Process Script — Language: Python, Theme: light, Mode: Default, Tab Size: 2

```
data = u"""[Risk Fabric]
<br>
Artifact User Account: {}
<br>
Risk Score: {}""".format(artifact.value, results.value)
incident.addNote(helper.createRichText(data))
```

- 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.


Workflows / RF Example: Mitigate Persistent Insider Threats
Mitigate Persistent Insider Threats
Persistent Insider Threat
RF Set Event Classifications
Notify internal management chain ...
Remote wipe/lock
RF Set Event Mitigations
Other
Input
Pre-Process Script
Output
Post-Process Script
- Font
+ Font

Name *: RF Example: Mitigate Persistent Insider Threats
API Name *: rf_example_mitigate_persistent_insider_threats
Description: Automatically Mitigate Persistent Insider Threats by Disabling Account
Object Type *: Incident

Pre-Process Script — Language: Python, Theme: light, Mode: Default, Tab Size: 2

```
inputs.rf_actionplanguid = incident.properties.rf_actionplanguid
inputs.rf_classification = "Violation"
```

Post-Process Script — Language: Python, Theme: light, Mode: Default, Tab Size: 2

```
data = u"""[Risk Fabric]
<br>
Events Classified: {}""".format(results.value.AffectedRecordCount)
incident.addNote(helper.createRichText(data))
```

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

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

Bay Dynamics Risk Fabric Function Version 1.0.0

Copyright © 2013 - 2019, Bay Dynamics, Inc. All rights reserved.

BAY DYNAMICS is a registered trademark of Bay Dynamics, Inc. Other names may be trademarks of their respective owners.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Bay Dynamics programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Bay Dynamics is not responsible for and expressly disclaims all warranties of any kind with respect to third-party content, products, and services. Bay Dynamics will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.