

IBM Resilient



Incident Response Platform Integrations

Carbon Black Protection Integration V1.0.2

Release Date: April 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 Carbon Black Protection Integration.

Overview

This integration consists of 12 functions which call various APIs to perform different actions, such as retrieving approval request details, updating approval requests and deleting files. It also contains a polling component to create incidents in the Resilient platform that correspond to approval requests in Carbon Black Protection.

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.
- Carbon Black Protection v8.1 or later.

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

```
sudo pip install --upgrade fn_cb_protection-1.0.2.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_<fn_name>] section, edit the settings as follows:

```
[fn_cb_protection]
# Name or IP address of your CbProtect server
server=10.200.1.1

# Access token issued by the CbProtect administrator
token= XXXX-XXXX-XXXX-XXXX

# If your CbProtect server has a self-signed TLS certificate, you cannot
verify it:
# verify_cert=false

# Interval (seconds) for automatic escalation of approval requests, set 0
to disable
# Suggest 300 as a starting point, which will check CbProtect every 5
minutes
escalation_interval=0

# Optional: query for which requests to escalate; default is to escalate
all open approval requests
# escalation_query=resolution:0

# Optional: path to a custom template file for the escalated incident
# template_tile=/usr/integration/bit9_escalation.jinja

# Optional: set this to only escalate a single request ID, e.g. when
testing a custom template
# test_single_request=999
```

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 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
```

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 package also includes example workflows and rules that show how the functions can be used. You can copy these workflows and rules for your own needs.













 Dashboards ▾ Simulations Incidents **Create**

  Resilient Sy...
TestOrg ▾

Functions

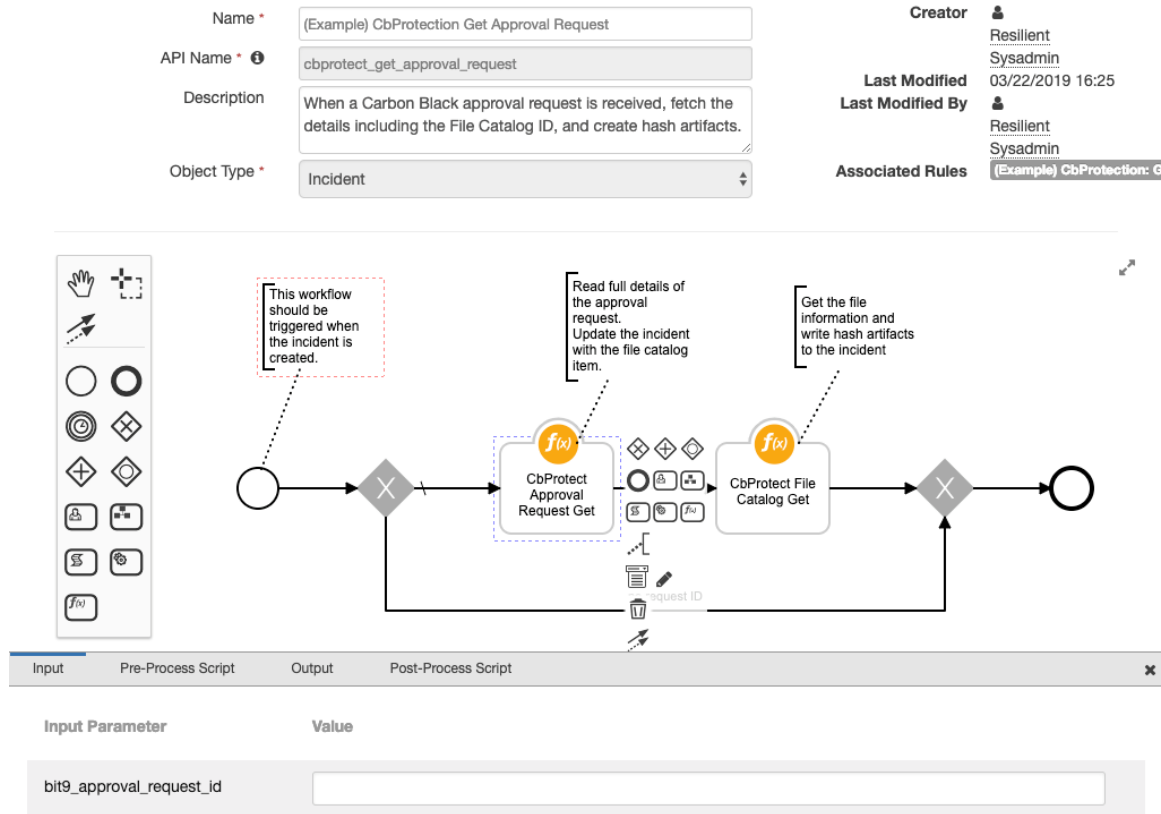
New Function



| Name | Description | |
|---|---|---|
| CbProtect Approval Request Get | Get an approval request by ID. |  |
| CbProtect Approval Request Query | Return approval requests that match the given criteria. |  |
| CbProtect Approval Request Update | Update an approval request. |  |
| CbProtect Delete File | Deletes a file from provided systems. |  |
| CbProtect File Catalog Get | Get a file catalog item by ID. |  |
| CbProtect File Catalog Query | Return file catalog objects that match the given criteria. |  |
| CbProtect File Instance Query | Return file instance objects that match the given criteria. |  |
| CbProtect File Instance Update | Update the approval state of a file instance. |  |
| CbProtect File Rule Delete | Delete a file rule. |  |
| CbProtect File Rule Get | Get a file rule by ID. |  |
| CbProtect File Rule Query | Return file rules that match the given criteria. |  |
| CbProtect File Rule Update | Create or update a File Rule. |  |

bit9_approval_request_get: Cbprotect Approval Request Get

Given an approval request's ID, the function returns the details of the approval request. The function takes one input, `bit9_approval-request_id`, which is a number. The following is an example of this function in the (Example) CbProtection Get Approval request workflow.



bit9_approval_request_query: Cbprotect Approval Request Query

This function takes one input, bit9_query which is a query sting, and returns the approval requests that match the given query. The following is an example of this function in the (Example) CbProtection Query Approval Request workflow:

[Workflows](#) / (Example) CbProtection Query Approval Request Cancel Save & Close Save

Name *

(Example) CbProtection Query Approval Request

API Name *

example_cbprotection_query_approval_request

Description

Queries for approval requests based on the provided query.

Object Type *

Incident

Creator

Resilient Sysadmin

Last Modified

03/25/2019 15:13

Last Modified By

Resilient Sysadmin

Associated Rules

(Example) CbProte

```
graph LR; Start(( )) --> Query[CbProtect Approval Request Query]; Query --> End(( ))
```

Input

Pre-Process Script

Output

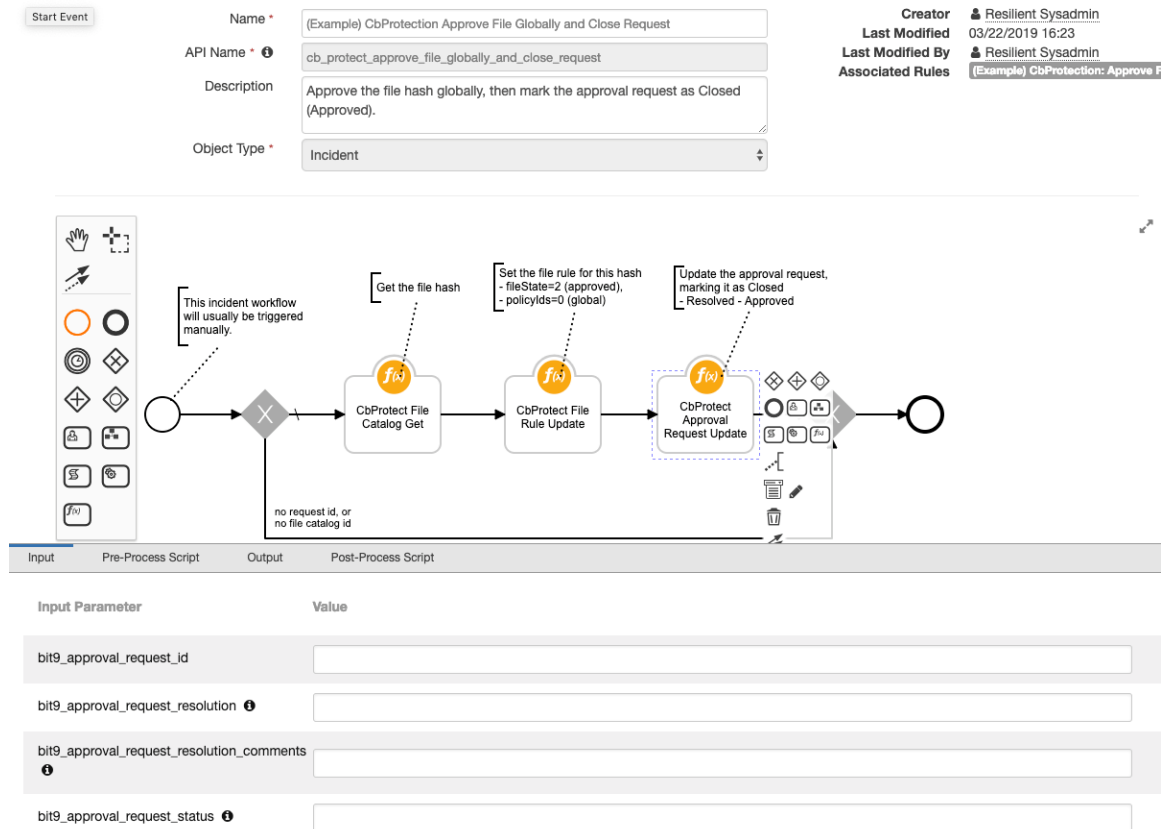
Post-Process Script

✕

| Input Parameter | Value |
|-----------------|----------------------|
| bit9_query | fileName:notepad.exe |

bit9_approval_request_update: Cbprotect Approval Request Update

This function accepts as input a request ID, approval request resolution, comments, and status. With these, it updates an approval request. The following is an example of this function in a workflow:



bit9_file_delete: Cbprotect Delete File

This function deletes a file from one or all computers using Carbon Black Protection. Set the `bit9_file_action` input to delete by file hash or file name (choose file hash to set catalog ID). Set the `bit9_computer_id` input for a specific computer, or use "0" to select all computers. Then, depending on the chosen file action, set the catalog ID, file hash, or file name. The following is an example of this function in the (Example) CbProtect Delete File workflow:

Name *

(Example) CbProtect Delete File

API Name *

example_cbprotect_delete_file

Description

Deletes a file from all systems.

Object Type *

Artifact

Creator Resilient Sysadmin

Last Modified 03/22/2019 14:08

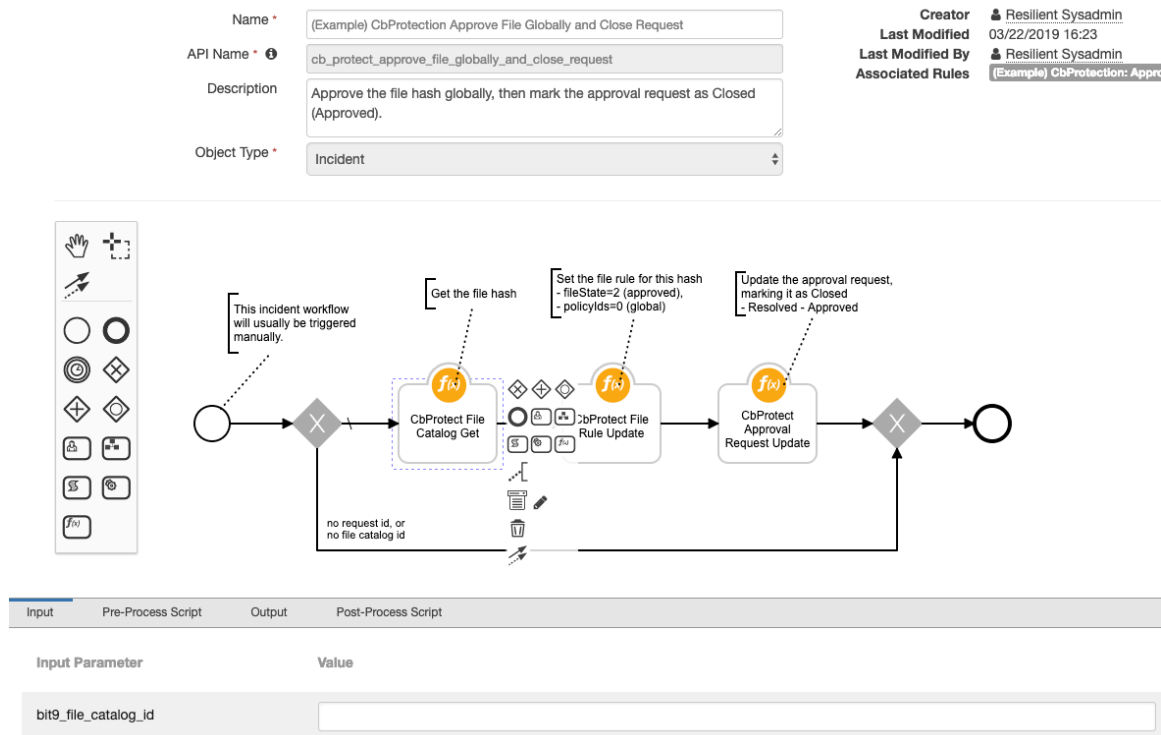
Last Modified By Resilient Sysadmin

Associated Rules (Example) CbProtect Delete File

| Input Parameter | Value |
|----------------------|---|
| bit9_file_action | <div style="border: 1px solid #ccc; padding: 2px;">DeleteFileByHash</div> |
| bit9_computer_id | <div style="border: 1px solid #ccc; height: 20px;"></div> |
| bit9_file_catalog_id | <div style="border: 1px solid #ccc; height: 20px;"></div> |
| bit9_file_hash | <div style="border: 1px solid #ccc; height: 20px;"></div> |
| bit9_file_name | <div style="border: 1px solid #ccc; height: 20px;"></div> |

bit9_file_catalog_get: Cbprotect File Catalog Get

Returns the file catalog details based on the catalog ID provided. The following is an example of this function in the (Example) CbProtection Approve File Globally and Close Request workflow:

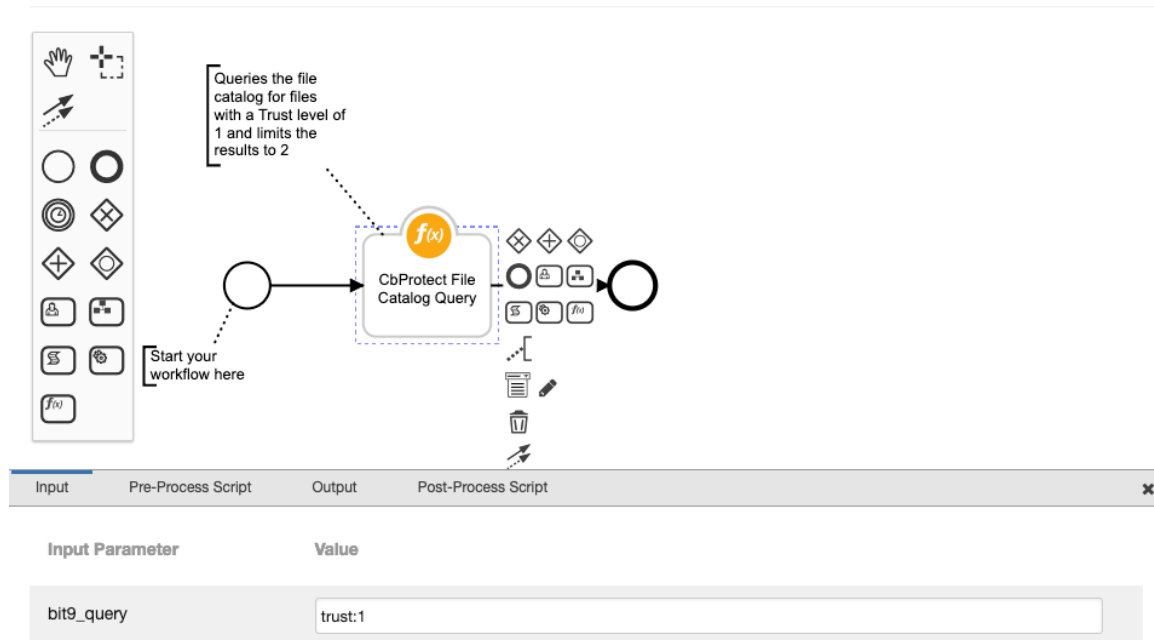


bit9_file_catalog_query: Cbprotect File Catalog Query

Returns file catalogs and their details from a provided query string. The following is an example of this function in the (Example) CbProtection Query File Catalog workflow:

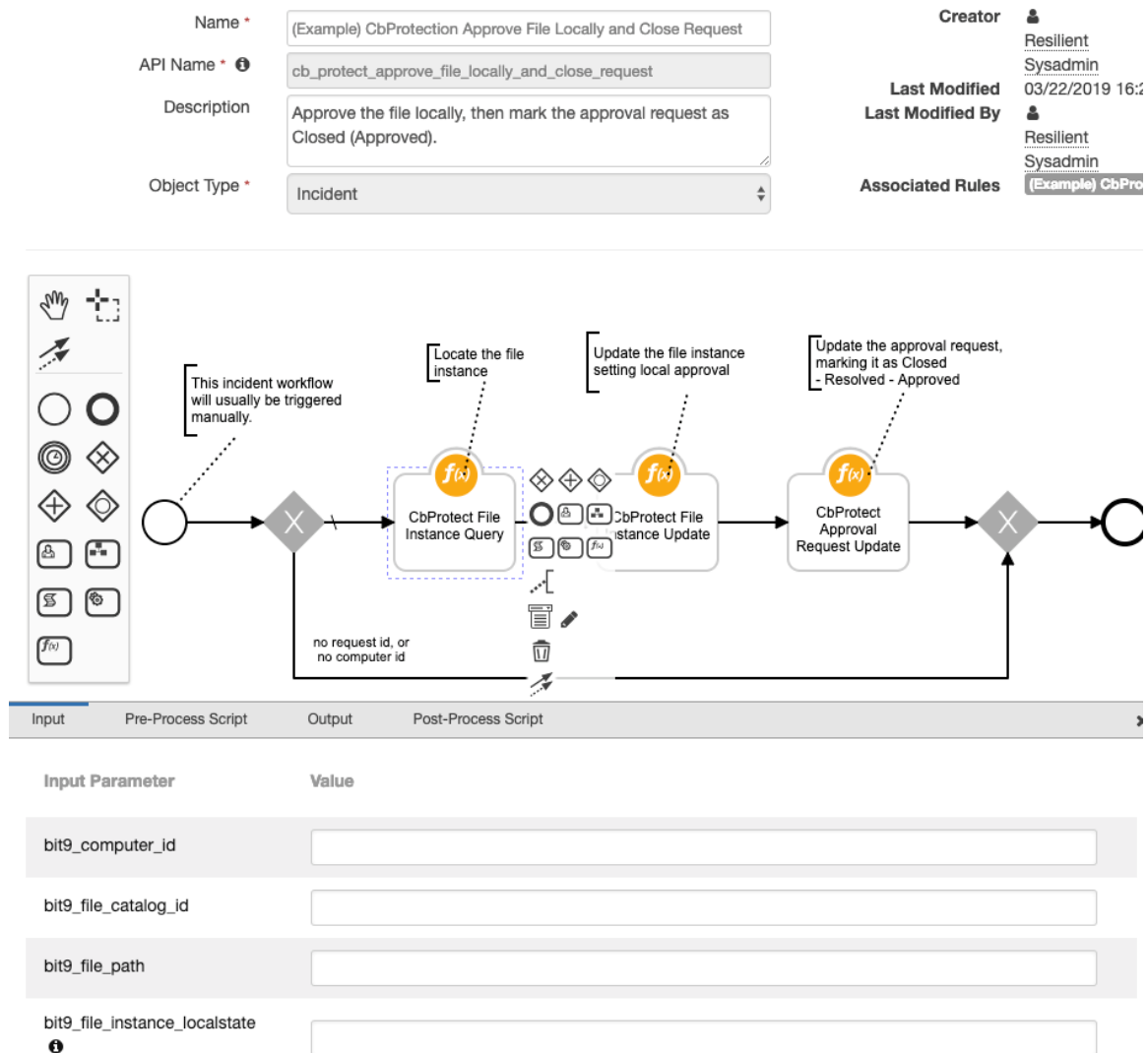
| | |
|---------------|--|
| Name * | (Example) CbProtection Query File Catalog |
| API Name * ⓘ | example_cbprotection_query_file_catalog |
| Description | Given a query, return details based on the file catalog results. |
| Object Type * | Incident |

| | |
|------------------|--------------------|
| Creator | Resilient Sysadmin |
| Last Modified | 03/21/2019 12:52 |
| Last Modified By | Resilient Sysadmin |
| Associated Rules | (Example) CbProte |



bit9_file_instance_query: Cbprotect File Instance Query

Returns file instance objects that match the given criteria from the inputs. The following is an example of this function in the (Example) CbProtection Approve File Locally and Close Request workflow:



```
graph LR; Start(( )) --> G1{X}; G1 --> Q[CbProtect File Instance Query]; Q --> U[CbProtect File Instance Update]; U --> A[CbProtect Approval Request Update]; A --> G2{X}; G2 --> End((( ))); G1 -- "no request id, or no computer id" --> End; Note1[This incident workflow will usually be triggered manually.] -.-> G1; Note2[Locate the file instance] -.-> Q; Note3[Update the file instance setting local approval] -.-> U; Note4[Update the approval request, marking it as Closed - Resolved - Approved] -.-> A;
```


| Input | Pre-Process Script | Output | Post-Process Script |
|-------------------------------|--------------------|--------|---------------------|
| bit9_computer_id | | | |
| bit9_file_catalog_id | | | |
| bit9_file_path | | | |
| bit9_file_instance_localstate | | | |

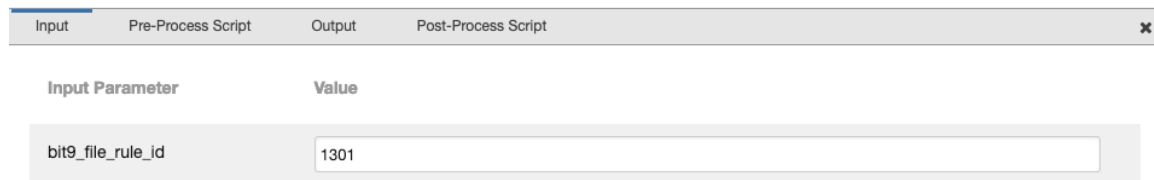
Updates a file instance's local approval/banned setting. This function has inputs for the file instance ID and the local state (for example, approved = 2). The following includes an example of this function in the (Example) CbProtection Approve File Locally and Close Request workflow:

The diagram illustrates a Business Process Automation (BPA) workflow for CbProtect File Instance Query and Update. The workflow starts with a manual trigger (represented by a hand icon) leading to a start node (circle). The process begins with a decision node (diamond with an 'X'). A callout box states: "This incident workflow will usually be triggered manually." The main path leads to the "CbProtect File Instance Query" task (rounded rectangle). A callout box points to this task: "Locate the file instance". The task output leads to the "CbProtect File Instance Update" task (rounded rectangle). A callout box points to this task: "Update the file instance setting local approval". The task output leads to the "CbProtect Approval request Update" task (rounded rectangle). A callout box points to this task: "Update the approval request, marking it as Closed - Resolved - Approved". The task output leads to a final decision node (diamond with an 'X'). A callout box points to this decision node: "no request id, or no computer id". The final decision node leads to an end node (circle).

Page 13


Given a file rule ID, deletes the file rule from Carbon Black. The following is an example of this function in the (Example) CbProtection Delete File Rule workflow:

| | |
|------------------|---|
| Creator |  <u>Resilient</u> Sysadmin |
| Last Modified | 03/20/2019 16:38 |
| Last Modified By |  <u>Resilient</u> Sysadmin |
| Associated Rules | (Example) CbProte |

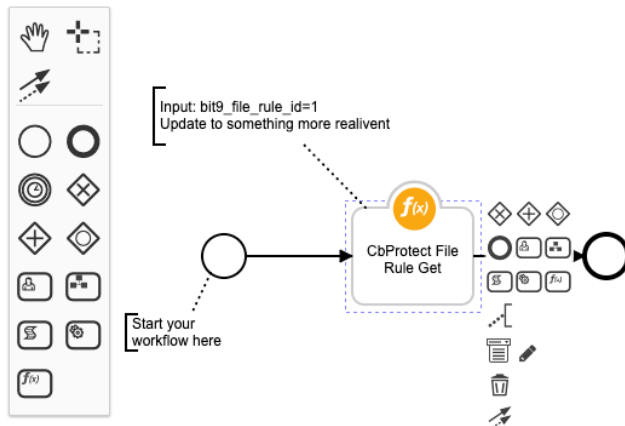


bit9_file_rule_get: Cbprotect File Rule Get

Given a file rule ID, returns the details of the file rule. The following is an example of this function in the (Example) CbProtection File Rule Get workflow:

| | |
|--|---|
| Name * | (Example) CbProtection File Rule Get |
| API Name *  | example_cbprotection_file_rule_get |
| Description | Given a file rule id, returns back the details of the rule. |
| Object Type * | Incident |



| | |
|------------------|---|
| Creator |  <u>Resilient</u> Sysadmin |
| Last Modified | 03/21/2019 10:25 |
| Last Modified By |  <u>Resilient</u> Sysadmin |
| Associated Rules | (Example) CbProte |

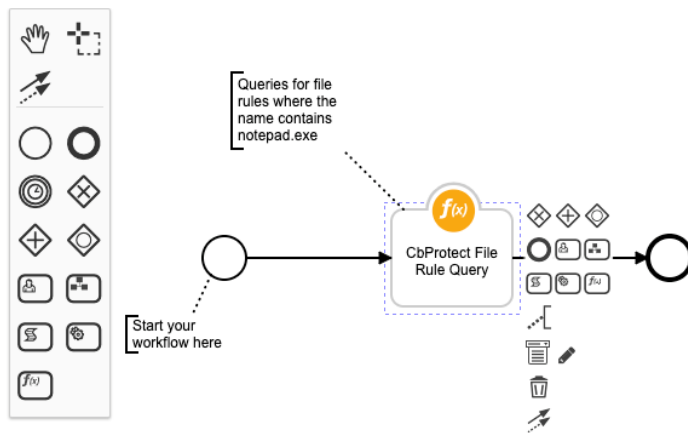


| Input | Pre-Process Script | Output | Post-Process Script |
|------------------------|--------------------|--------|---------------------|
| Input Parameter | | | |
| Value | | | |
| bit9_file_rule_id | | 1 | |

bit9_file_rule_query: Cbprotect File Rule Query

Given a query string, returns details of the file rules that match the query. The following is an example of this function in the (Example) CbProtection Query File Rule workflow:

| | | | |
|---------------|--|------------------|--|
| Name * | (Example) CbProtection Query File Rule | Creator |  Sysadmin |
| API Name * | example_cbprotection_query_file_rule | | |
| Description | Given a query, returns back the details of the file rules which match the query. | Last Modified | 03/21/2019 10:25 |
| | | Last Modified By |  Resilient Sysadmin |
| Object Type * | Incident | Associated Rules | (Example) CbProte |



| Input | Pre-Process Script | Output | Post-Process Script | |
|------------------------|---|--------|---------------------|--|
| Input Parameter | | | | |
| Value | | | | |
| bit9_query | <input type="text" value="name:notepad.exe"/> | | | |

bit9_file_rule_update: Cbprotect File Rule Update

This function updates a file rule in Carbon Black based on the data passed as inputs. The following is an example of this function in the (Example) CbProtection Approve File Globally and Close Request workflow:

Name *

(Example) CbProtection Approve File Globally and Close Request

API Name *

cb_protect_approve_file_globally_and_close_request

Description

Approve the file hash globally, then mark the approval request as Closed (Approved).

Object Type *

Incident

Creator

Resilient Sysadmin

Last Modified

03/22/2019 16:23

Last Modified By

Resilient Sysadmin

Associated Rules

(Example) CbProte

Input

Pre-Process Script

Output

Post-Process Script

bit9_file_rule_id

bit9_file_catalog_id

bit9_file_rule_name

bit9_file_rule_description

bit9_file_rule_filestate ⓘ

bit9_file_rule_sourcetype ⓘ

bit9_file_rule_policyids ⓘ

bit9_file_rule_hash

Carbon Black Protection Resilient Polling Component

This integration contains a polling component that automatically escalates approval requests into the Resilient platform. To enable this feature, the `escalation_interval` variable in the `app.config` file must be set to an integer greater than 0. This integer represents the interval in number of seconds for the automatic escalation of approval requests. It is recommended to start at 300, which checks every 5 mins.

You can also set optional values, such as `escalation_query`, which escalates approval requests that match the query; if not set it defaults to all open approval requests. In addition, you can set `template_file` to the location of a custom jinja template file; if not set, the default template file is used. To create your own custom jinja file, you should use the default jinja file as a reference. This file can be found when expanding the package in the following directory:

```
fn_cb_protection-<version#>/fn_cb_protection/data/
```

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.