



# Incident Response Platform Integrations

# X-Force Collections Function V1.0.0

Release Date: <month> 2018

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 X-Force Collections Function.

Overview

IBM X-Force Exchange is a threat intelligence sharing platform enabling research on security threats, aggregation of intelligence, and collaboration with peers.

These functions allow users to search X-Force Exchange Collections for enrichment information about a threat. Users may search all collections using an IP Address, File Hash, DNS Record or a string as a query which will return a number of collections which match the query. Users can also get a specific collection by its identifier which will return information about the requested collection to the Resilient Platform for use in other functions.

Installation

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

* Resilient platform is version 30 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 the command line of the Resilient appliance, which hosts the Resilient platform; or to a separate integration server where you will deploy and run the functions code. If using a separate integration server, you must install Python version 2.7.10 or later, or version 3.6 or later, and “pip”. (The Resilient appliance is preconfigured with a suitable version of Python.)

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\_xforce-1.0.0.zip

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\_xforce] section, edit the settings as follows:

xforce\_apikey = <YOUR\_API\_KEY>

xforce\_password = <YOUR\_API\_PASSWORD>

xforce\_https\_proxy = <YOUR\_PROXY\_URL>

xforce\_http\_proxy = <YOUR\_PROXY\_URL>

xforce\_base\_url = <XFORCE\_BASE\_URL>

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

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:

[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:

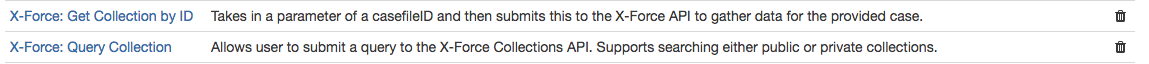
sudo systemctl resilient\_circuits [start|stop|restart|status]

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 Descriptions

Once the function package deploys the function(s), 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.



X-Force: Get Collection by ID:

This function takes in a parameter of a collectionID to be searched. The function then creates a connection to the X-Force REST API and attempts to gather collection information using the provided ID. The return result is saved as a note in this example workflow.

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Example** |
| xforce\_collection\_id | String identifier for a collection | “a1a447c6c22b6554258cbf0e2bba1a0d” |

X-Force: Query Collection(s):

This function allows a user to submit a query in the form of a String, IP Address, Hash, DNS entry which is then used to search either the public or private collection in X-Force Exchange. The function creates a connection to the X-Force REST API and submits the query also specifying whether public or private results are desired. The result of this is a number of matching collections which each have a collectionID.

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Example** |
| xforce\_query | String query which may be an IP,DNS,File hash or just a string | “Mirai” |
| xforce\_collection\_type | String specifying whether to search in public or private collections | “public” |

X-Force: Return Top 3 from Collection(s)

This workflow combines the first two defined functions to create a workflow which will allow a user to submit a query in the form of a String, IP Address, Hash, DNS entry which is then used to search either the public or private collection in X-Force Exchange. Results, which come in the form of an array of strings, will be then passed downstream to the next part of the workflow. The top 3 results received from the output of the query will be submitted to X-Force to gather enrichment information about each one to display in the Resilient Platform as a Note.

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](mailto:support@resilientsystems.com).

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