IBM Resilient

Incident Response Platform Integrations

Cisco Umbrella Investigate Function V1.0.3

Version	Publication	Notes
1.0.3	September 2020	Proxy Support
1.0.2	May 2020	App Host support added
1.0.1	August 2019	Configuration settings checking less restrictive.
1.0.0	June 2018	Initial publication.

Resilient Functions simplify development of integrations by wrapping each activity into an individual workflow component. These components can be easily installed and then used and combined in Resilient workflows. The Resilient platform sends data to the function component that performs an activity and then returns the results to the workflow. The results can be actioned by scripts, rules, and workflow decision points to dynamically orchestrate the security incident response activities.

This guide describes the Cisco Umbrella Investigate Function.

Overview

Umbrella Investigate is the interface to the security data collated by the Cisco Umbrella Investigate research team. The Cisco Umbrella Investigate REST API service allows for the querying of the Umbrella DNS database to show security events and correlations in their datasets. The Investigate REST API opens up the power of the Investigate classification results, correlation, and history, and is based on the Umbrella global network, the world's largest security network.

The Cisco Umbrella Investigate integration with the Resilient platform allows querying of the Investigate datasets using their REST APIs. The returned results can be used to make customized updates to the Resilient platform, such as updating incidents, artifacts, data tables and so on.

There are 14 functions supplied in the Resilient Function package for Umbrella Investigate. The functions interrogate the various REST APIs exposed by the Investigate service. There are also example workflows in the customizations section of the package which demonstrate usage of the Resilient Investigate Functions to update data tables.

The remainder of this document describes the included functions, how to configure example custom workflows, and data tables.

Installation

App Host

All the components for running this integration in a container already exist when using the App Host app.

To install,

- Navigate to Administrative Settings and then the Apps tab.
- Click the Install button and select the downloaded file: app-fn_cisco_umbrella_inv-x.x.x.zip.
- Go to the Configuration tab and edit the app.config file, editing the api_token for Cisco Umbrella access and adjusting any other setting as needed.

```
[fn_cisco_umbrella_inv]
base_url=https://investigate.api.umbrella.com/
# The api_token will be supplied by Cisco will be in uuid format.
api_token=<api token>
results_limit=200
# uncomment to specify proxy settings
#https_proxy=https://your.proxy.com
#http_proxy=http://your.proxy.com
```

Integration Server

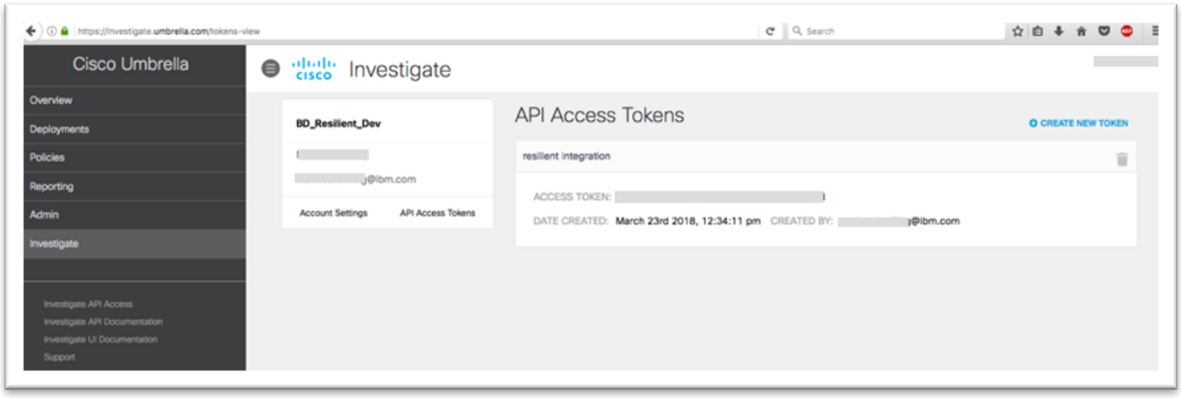
Before installing, verify that your environment meets the following prerequisites:

- Resilient platform must be version 30 or later.
- You must 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 must 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 you are 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.)

Configure Cisco Umbrella Investigate

The Umbrella Investigate default base URL is https://investigate.api.umbrella.com/. You can override the base URL if required.

Access to the Cisco Umbrella Investigate REST API is allowed by providing an access token in the request. The access token is tied to a user account on the Umbrella platform.



More information is available at https://investigate-api.readme.io/docs/about-the-api-authentication.

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. Run the following command to install the package:

```
sudo pip install --upgrade fn_cisco_umbrella_inv-1.0.1.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_cisco_umbrella_inv] section, edit the settings as follows:

```
base_url=https://investigate.api.umbrella.com/
```

```
# The api_token will be supplied by Cisco will be in uuid format.
api_token=abcd1234-a123-123a-123a-123456abcdef
results_limit=200
```

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, using 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 recommended way to do this is to configure it to automatically run at start up. On a Red Hat appliance, you can do this using a `systemd` unit file such as the one below. You might 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.lo
ck

[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:

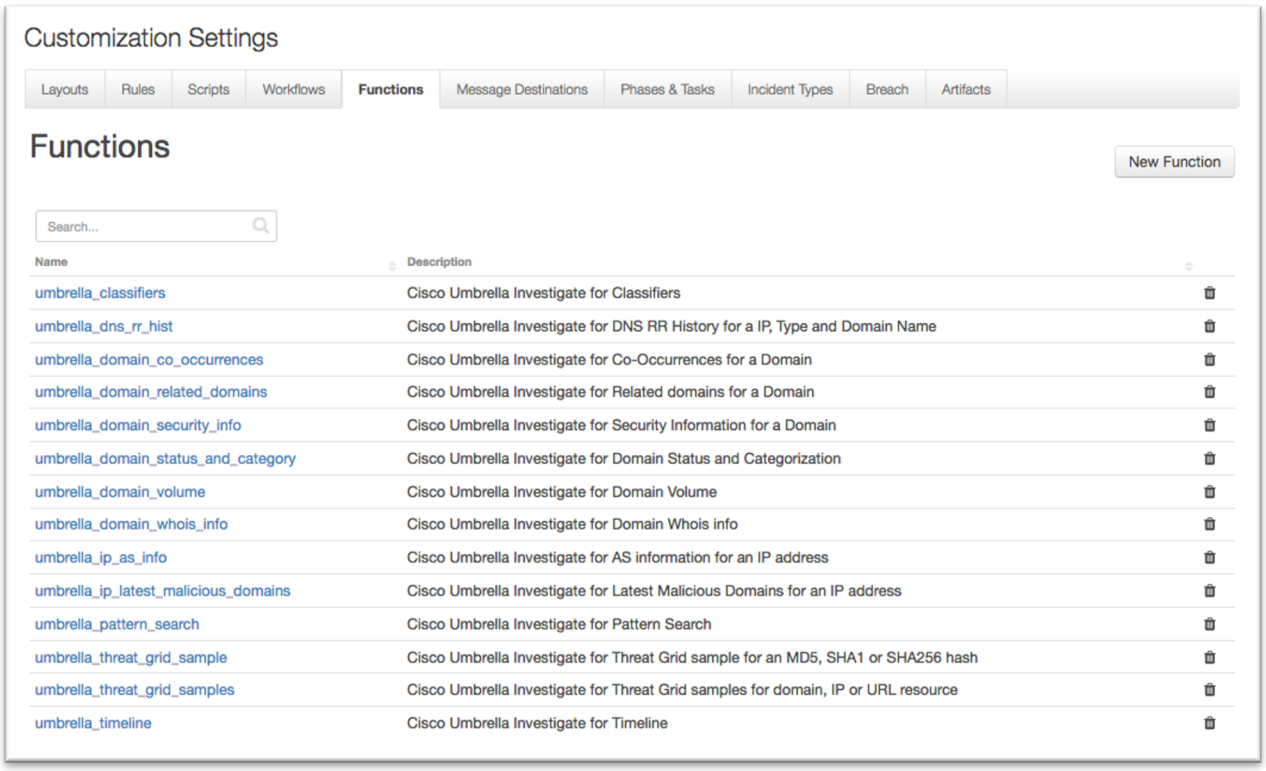
```
sudo systemctl resilient_circuits [start|stop|restart|status]
```

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.



Functions and Components

The package includes example workflows, rules and data tables that show how you can use the functions, as shown in the following table. Resilient users can view the rules in the Rules tab and the workflows in the Workflows tab, and modify them as needed. The object type for the workflows is Artifact, with the exception of the workflow, **Example: Get list of category identifiers**, which has an Incident object type.

Function	Rule	Workflow	Data Table
umbrella_ip_as_info	Example: AS Information for an ip address or ASN	Example: AS Information for an ip address or ASN	Umbrella Investigate - AS Information for an ip address or ASN
umbrella_domain_status_and_category	Example: Get list of category identifiers	Example: Get list of category identifiers	Umbrella Investigate - Category identifiers
umbrella_domain_status_and_category	Example: Categories for a domain	Example: Categories for a domain	Umbrella Investigate - Categories for a domain
umbrella_classifiers	Example: Classifiers for a domain	Example: Classifiers for a domain	Umbrella Investigate - Classifiers for a domain

Function	Rule	Workflow	Data Table
umbrella_domain_co_ occurrences	Example: DNS RR history for a domain	Example: DNS RR history for a domain	Umbrella Investigate - Co-occurrences for a domain
umbrella_dns_rr_hist	Example: DNS RR history for a domain	Example: DNS RR history for a domain	Umbrella Investigate - DNS RR history for a domain
umbrella_dns_rr_hist	Example: DNS RR history for an ip address	Example: DNS RR history for an ip address	Umbrella Investigate - DNS RR history for an ip address
umbrella_domain_volume	Example: Domain volume	Example: Domain volume	Umbrella Investigate - Domain Volume
umbrella_domain_whois_ info	Example: Domain WHOIS information for a domain	Example: Domain WHOIS information for a domain	Umbrella Investigate - Domain WHOIS info for a domain
umbrella_ip_latest_ malicious_domains	Example: Latest Malicious Domains for an ip address	Example: Latest Malicious Domains for an ip address	Umbrella Investigate - Latest Malicious Domains for an IP
umbrella_pattern_search	Example: Pattern search start epoch	Example: Pattern search start epoch	Umbrella Investigate - Pattern search with start epoch
umbrella_pattern_search	Example: Pattern search start relative	Example: Pattern search start relative	Umbrella Investigate - Pattern search with start relative
umbrella_domain_related _domains	Example: Related Domains for a Domain	Example: Related Domains for a Domain	Umbrella Investigate - Related Domains for a Domain
umbrella_domain_security _info	Example: Security information for a domain	Example: Security information for a domain	Umbrella Investigate - Security information for a domain
umbrella_threat_grid_ sample	Example: ThreadGrid sample information for a hash	Example: ThreadGrid sample information for a hash	Umbrella Investigate - ThreadGrid sample info for a hash
umbrella_threat_grid_ samples	Example: ThreadGrid samples for a resource	Example: ThreadGrid samples for a resource	Umbrella Investigate - ThreadGrid samples for a resource
umbrella_timeline	Example: Timeline for a resource	Example: Timeline for a resource	Umbrella Investigate - Timeline for a resource

Inputs

Each function has a set of inputs, which you can view by clicking the function name in the Functions tab of the Resilient platform.

The Resilient functions use input parameters starting with `umbinv_`, examples include `umbinv_domains`, `umbinv_showlabels` and `umbinv_status_endpoint`. These are equivalent to the input parameters, endpoints, and qualifiers used in the REST API call. Refer to Introduction to Cisco Umbrella Investigate and the Cisco Umbrella API documentation on the use of the Umbrella Investigate inputs.

The following input parameter is used in the Resilient functions where the input can be one of several different types.

```
umbinv_resource e.g. ip_address, email_address or nameserver
```

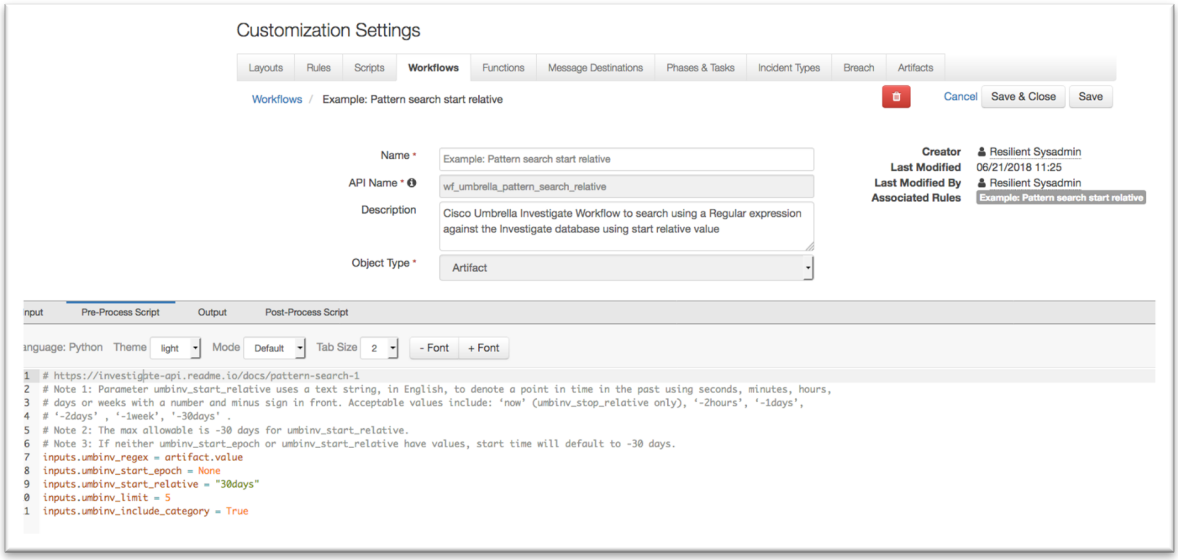
The stop and start input parameter values in the REST API are derived from a pair of mutually exclusive equivalent parameters defined in the Resilient functions.

```
unbinv_start_epoch or unbinv_stop_epoch
umbinv_start_relative or umbinv_stop_relative
```

Parameters `umbinv_start_epoch` and `umbinv_stop_epoch` use a Datepicker to populate the value in the Workflow Input tab. This value is translated to a Unix epoch timestamp in milliseconds, by the Resilient platform. The Workflow Pre-Process Script accepts a Unix timestamp value in miliseconds.

Parameters `umbinv_start_relative` and `umbinv_stop_relative` use a text string, in English, to denote a point in time in the past using seconds, minutes, hours, days or weeks with a number and minus sign in front. Acceptable values include: 'now' (`umbinv_stop_relative` only), '-2hours', '-1days', '-2days', '-1week'. Parameter `umbinv_start_relative` must be <= parameter `umbinv_stop_relative`. The max allowable value is -30 days.

Resilient users can set the values for the inputs when editing the function within one of the provided workflows. The user can change the value in the Input or Pre-process Script tab. The following is an example of a function's Pre-Process Script tab within a workflow.

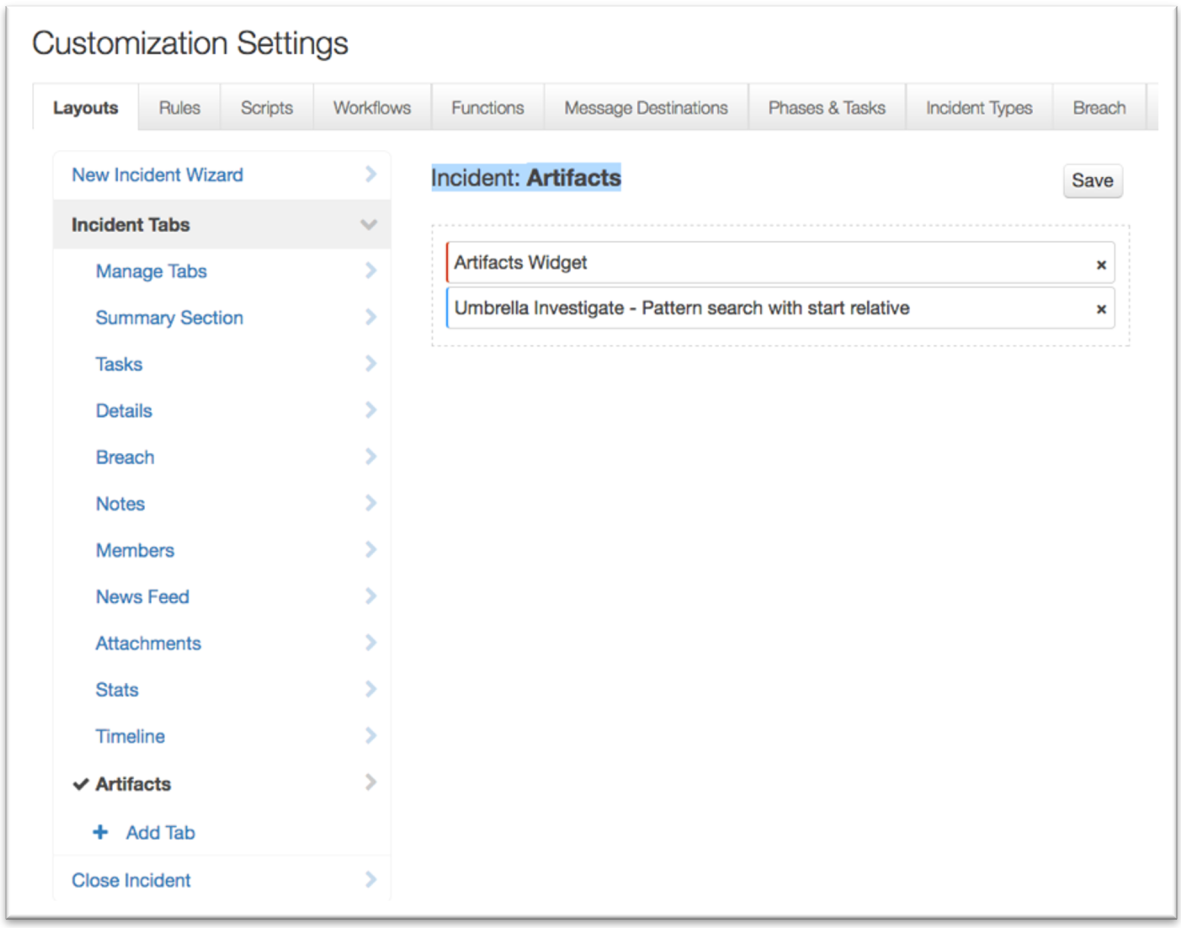


Resilient Platform Configuration

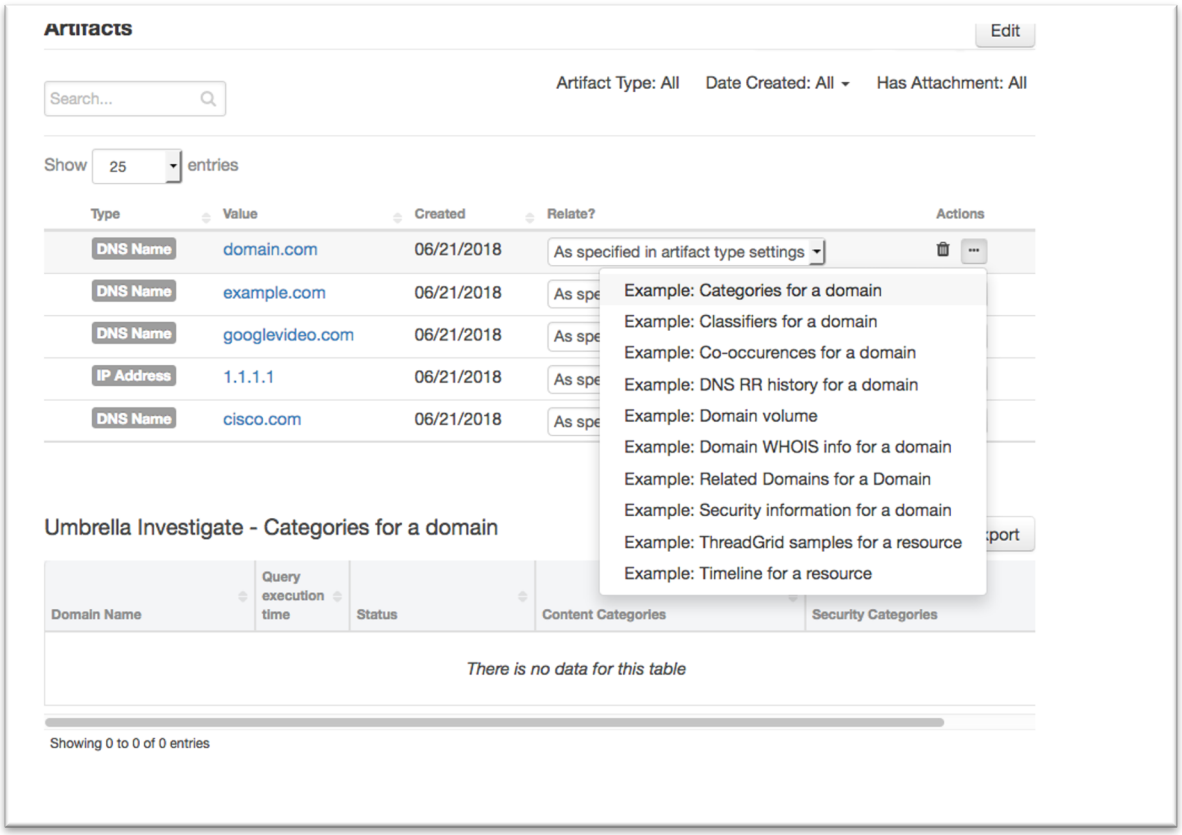
To display results, users need to manually add the provided data tables to the Artifacts tab, as follows:

1. Navigate to the Customization Settings and select the Layouts tab.
2. Select Artifacts.
3. Drag each data table to your Artifacts tab.
4. Click **Save**.

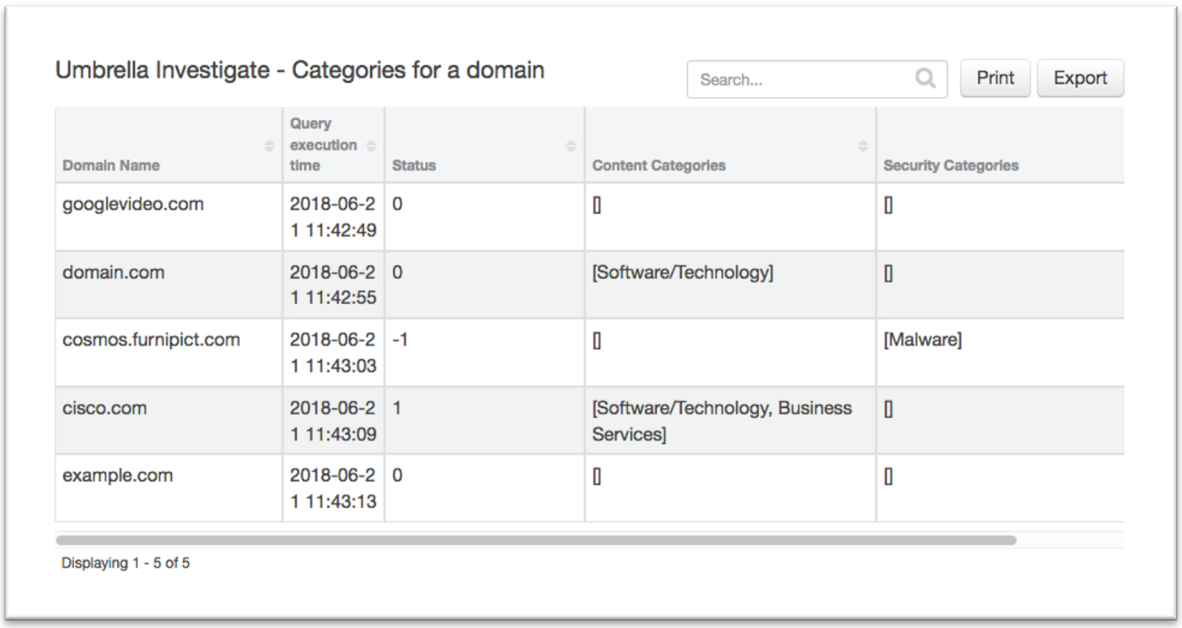
The following screenshot shows the **Umbrella Investigate - Pattern search with start relative** data table added to Artifacts tab.



Users can run a Cisco Umbrella Investigate query by clicking the Actions icon for an artifact, then selecting a rule. This executes the corresponding workflow against that particular artifact. In the following example, when a user executes the rule, **Example: Categories for a domain**, the corresponding data table is updated, where the artifact values are domain names.



Some workflows add more than one row per artifact for each execution. For example:



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.