Cisco ASA

Table of Contents

Release Notes

Version	Date	Notes
1.2.0	04/2023	Convert to playbooks
1.1.0	06/2022	Update payload samples
1.0.0	04/2021	Initial Release

Overview

IBM QRadar SOAR app for Cisco ASA

The Cisco ASA Family of security devices protects corporate networks and data centers of all sizes. It provides users with highly secure access to data and network resources - anytime, anywhere, using any device. Cisco ASA devices represent more than 15 years of proven firewall and network security engineering and leadership, with more than 1 million security appliances deployed throughout the world.

Cisco Adaptive Security Appliance (ASA) Software is the core operating system for the Cisco ASA Family. It delivers enterprise-class firewall capabilities for ASA devices in an array of form factors - standalone appliances, blades, and virtual appliances - for any distributed network environment. ASA Software also integrates with other critical security technologies to deliver comprehensive solutions that meet continuously evolving security needs.

Cisco ASA firewalls are historically managed through the command line, however they do provide a robust REST API for integrating with 3rd party products. The IBM Security SOAR Cisco ASA app uses the Cisco ASA REST API to allow SOC analyst to controll internet access of machines from the IBM Security SOAR Platform.

Key Features

Key capabilities include the following:

- Allows a SOC analyst to pre-configure available firewalls with credentials in the app.config file. Each firewall contains a list of Cisco ASA named network object groups for blocking inbound traffic and outbound traffic, also specified in the app.config.
- Provides the ability to display all IP addresses currently in a network object group blocklist in a data table.
- Provides the ability to add IP address to the blocklist (network object group).
- Provides the ability to remove an IP addresses from blocklist (network object group).
- The following IP network objects are can be added/removed from a network object group:
  - IPv4Address
  - IPv4Range
  - IPv4Network
  - IPv4FQDN
  - IPv6Address
  - IPv6Range
  - IPv6Network
  - IPv6FQDN

Requirements

This app supports the IBM Security QRadar SOAR Platform and the IBM Security QRadar SOAR for IBM Cloud Pak for Security.

SOAR platform

The SOAR platform supports two app deployment mechanisms, Edge Gateway (formerly App Host) and integration server.

If deploying to a SOAR platform with an Edge Gateway, the requirements are:

- SOAR platform >= `46.0.8131`.
- The app is in a container-based format (available from the AppExchange as a `zip` file).

If deploying to a SOAR platform with an integration server, the requirements are:

- SOAR platform >= `xv`.

- The app is in the older integration format (available from the AppExchange as a `zip` file which contains a `tar.gz` file).
- Integration server is running `resilient-circuits>=48.0.0`.
- If using an API key account, make sure the account provides the following minimum permissions:

Name	Permissions
Org Data	Read
Org Data	Edit
Function	Read
Incidents	Read
Edit Incidents	Fields

The following SOAR platform guides provide additional information:

- *Edge Gateway Deployment Guide* or *App Host Deployment Guide*: provides installation, configuration, and troubleshooting information, including proxy server settings.
- *Integration Server Guide*: provides installation, configuration, and troubleshooting information, including proxy server settings.
- *System Administrator Guide*: provides the procedure to install, configure and deploy apps.

The above guides are available on the IBM Documentation website at ibm.biz/soar-docs. On this web page, select your SOAR platform version. On the follow-on page, you can find the *Edge Gateway Deployment Guide*, *App Host Deployment Guide*, or *Integration Server Guide* by expanding **Apps** in the Table of Contents pane. The System Administrator Guide is available by expanding **System Administrator**.

Cloud Pak for Security

If you are deploying to IBM Cloud Pak for Security, the requirements are:

- IBM Cloud Pak for Security >= `1.8`.
- Cloud Pak is configured with an Edge Gateway.
- The app is in a container-based format (available from the AppExchange as a `zip` file).

The following Cloud Pak guides provide additional information:

- *Edge Gateway Deployment Guide* or *App Host Deployment Guide*: provides installation, configuration, and troubleshooting information, including proxy server settings. From the Table of Contents, select Case Management and Orchestration & Automation > **Orchestration and Automation Apps**.
- *System Administrator Guide*: provides information to install, configure, and deploy apps. From the IBM Cloud Pak for Security IBM Documentation table of contents, select Case Management and Orchestration & Automation > **System administrator**.

These guides are available on the IBM Documentation website at ibm.biz/cp4s-docs. From this web page, select your IBM Cloud Pak for Security version. From the version-specific IBM Documentation page, select Case Management and Orchestration & Automation.

Proxy Server

The app **does** support a proxy server.

Python Environment

Python 3.6 and Python 3.9 are supported. Additional package dependencies may exist for each of these packages:

- resilient-circuits>=48.0.0

Development Version

This app has been implemented using:

Product Name	Product Version	API URL	API Version
Cisco ASAv	9.14(1)	https://host	N/A

Installation

Install

- To install or uninstall an App or Integration on the *SOAR platform*, see the documentation at ibm.biz/soar-docs.
- To install or uninstall an App on *IBM Cloud Pak for Security*, see the documentation at ibm.biz/cp4s-docs and follow the instructions above to navigate to Orchestration and Automation.

App Configuration

The following table provides the settings you need to configure the app. These settings are made in the app.config file. See the documentation discussed in the Requirements section for the procedure.

The app.config file for this app contains a high level section denoted by `[fn_cisco_asa]` and subsections for each firewall denoted as `[fn_cisco_asa:firewall_name]`, where each firewall_name is unique.

The table below provides the optional high level default settings. The credentials defined in this section are the default credentials used if the credentials are not defined in the individual firewall subsection.

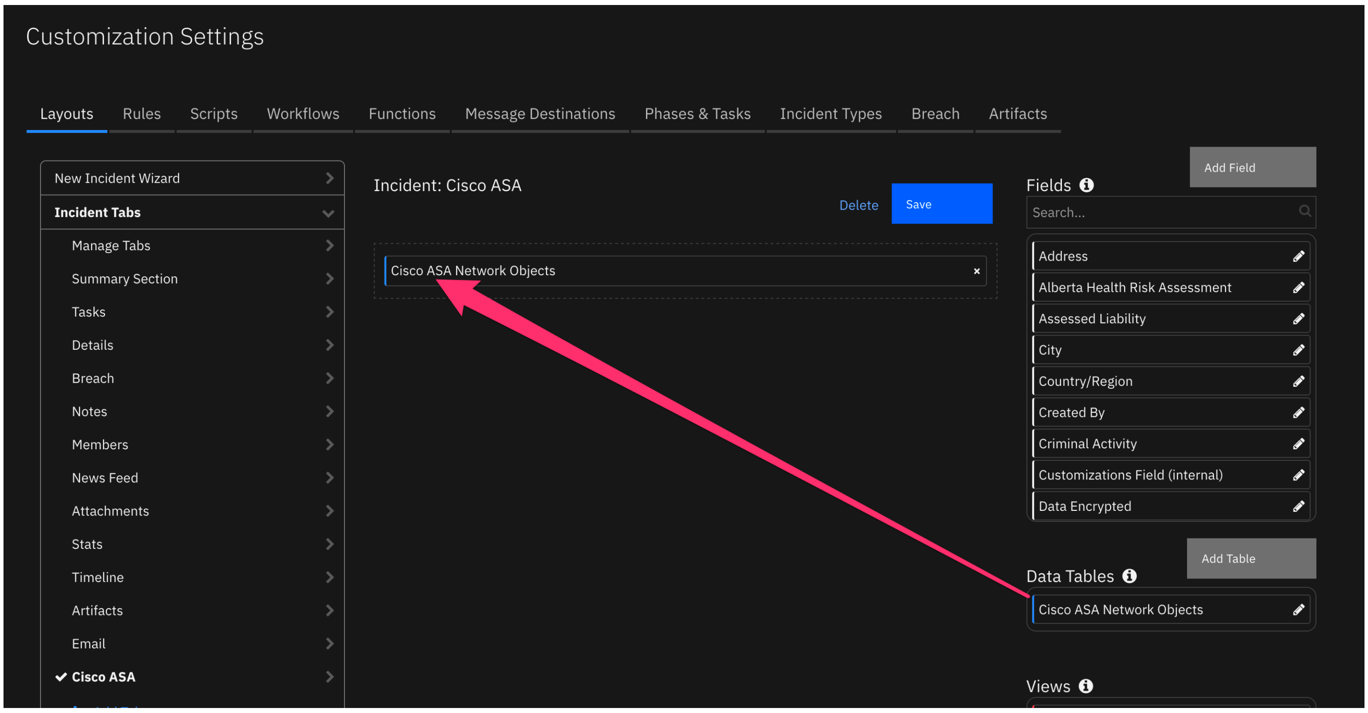
Config	Required	Example	Description
username	No	`<asa_username>`	*Username of the Cisco ASA firewall*
password	No	`<asa_password>`	*Password of the Cisco ASA firewall.*
https_proxy	No	`https://your.proxy.com`	-
http_proxy	No	`http://your.proxy.com`	-

Config	Required	Example	Description
host	Yes	`<asa_ip>`	*IP Address of the Cisco ASA firewall.*
username	No	`<asa_username>`	*Username of the Cisco ASA firewall*

Config	Required	Example	Description
password	No	`<asa_password>`	*Password of the Cisco ASA firewall.*
network_object_groups	Yes	`BLOCKLIST_IN, BLOCKLIST_OUT`	*Comma separated list of the Cisco ASA network object groups.*
cafile	No	-	*Path to certificate file.*

Custom Layouts

Import the Cisco ASA Network Objects Data Tables and drag it onto a Cisco ASA Incident tab as shown in the screenshot below:



Cisco ASA Configuration

Install and Configure ASA REST API Agent and Client

To run the Cisco ASA app, you must first install and configure the Cisco ASA REST API Agent and Client on each device as described in the Cisco ASA REST API Quick Start Guide.

Create Cisco ASA Network Object Groups

The network object groups defined in the app.config are created by a user before running the app. The Cisco ASA CLI (command line interface) or the ASDM (Cisco Adaptive Security Device Manager - GUI interface) can be used to create the network object groups.

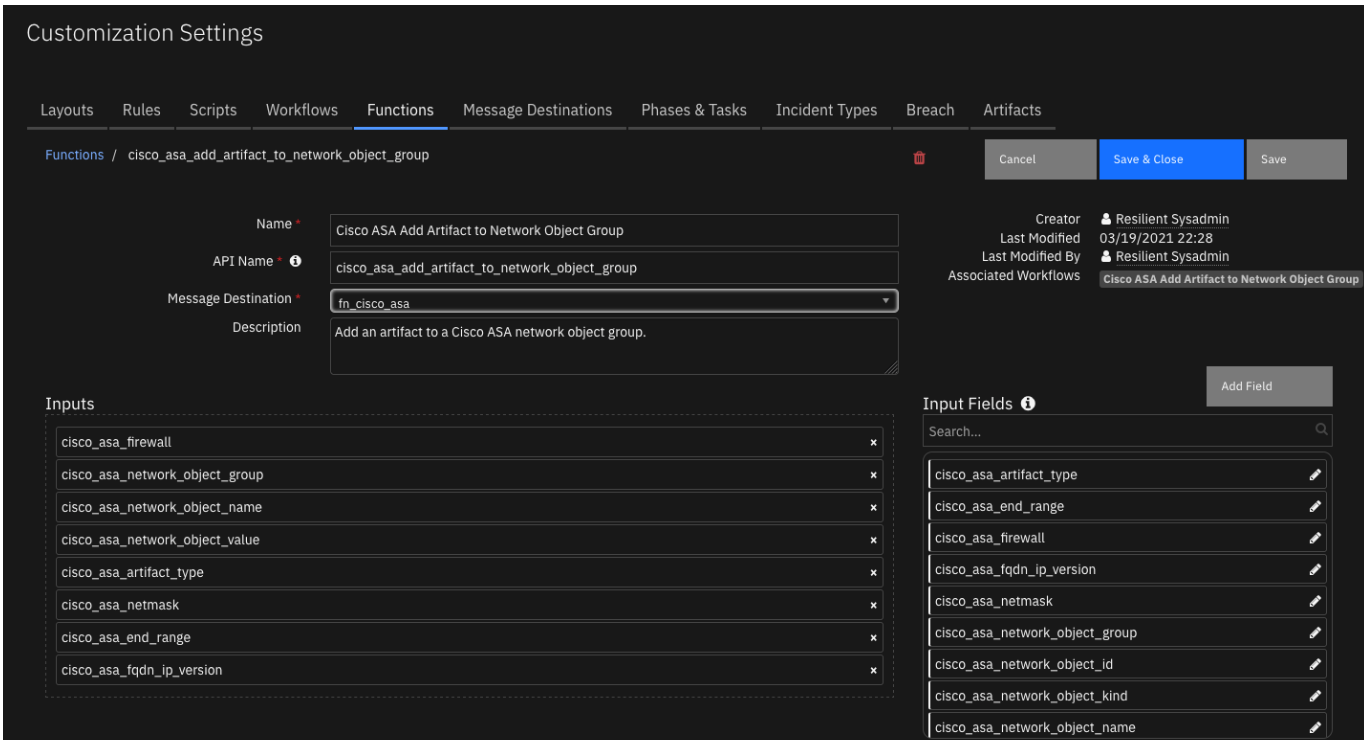
Here is an example configuration to create a network object group called BLOCKLIST_IN using the CLI:

```
hostname(config)# object-group network BLOCKLIST_IN
hostname(config-network)# network-object host 192.168.10.1
hostname(config-network)# network-object host 192.168.10.2
hostname(config-network)# network-object host 192.168.10.3
hostname(config-network)# access-list my-internet-access deny ip object-
group BLOCKLIST_IN any
hostname(config)# access-list my-internet-access permit ip any any
hostname(config)# access-group my-internet-access in interface inside
```

The app makes REST API calls to add and remove network objects from the BLOCKLIST_IN network object group.

Function - Cisco ASA Add Artifact to Network Object Group

Add an artifact to a Cisco ASA network object group.



Functions / cisco_asa_add_artifact_to_network_object_group

Name *	Cisco ASA Add Artifact to Network Object Group
API Name *	cisco_asa_add_artifact_to_network_object_group
Message Destination *	fn_cisco_asa
Description	Add an artifact to a Cisco ASA network object group.
Creator	Resilient Sysadmin
Last Modified	03/19/2021 22:28
Last Modified By	Resilient Sysadmin
Associated Workflows	Cisco ASA Add Artifact to Network Object Group

Inputs

- cisco_asa_firewall
- cisco_asa_network_object_group
- cisco_asa_network_object_name
- cisco_asa_network_object_value
- cisco_asa_artifact_type
- cisco_asa_netmask
- cisco_asa_end_range
- cisco_asa_fqdn_ip_version

Input Fields

Search...

- cisco_asa_artifact_type
- cisco_asa_end_range
- cisco_asa_firewall
- cisco_asa_fqdn_ip_version
- cisco_asa_netmask
- cisco_asa_network_object_group
- cisco_asa_network_object_id
- cisco_asa_network_object_kind
- cisco_asa_network_object_name

▶ Inputs:

Name	Type	Required	Example	Tooltip
`cisco_asa_artifact_type`	`text`	Yes	`-`	-
`cisco_asa_end_range`	`text`	No	`-`	-
`cisco_asa_firewall`	`text`	Yes	`-`	-
`cisco_asa_fqdn_ip_version`	`select`	No	`-`	-
`cisco_asa_netmask`	`text`	No	`-`	-
`cisco_asa_network_object_description`	`text`	No	`-`	-
`cisco_asa_network_object_group`	`text`	Yes	`-`	-
`cisco_asa_network_object_name`	`text`	No	`-`	-
`cisco_asa_network_object_value`	`text`	Yes	`-`	-

▶ Outputs:

> **NOTE:** This example might be in JSON format, but `results` is a Python Dictionary on the SOAR platform.

```
results = {
  "content": {
    "firewall": "firewall_1",
    "network_object_description": "fqdn name",
    "network_object_group": "BLOCKLIST_IN",
    "network_object_kind": "IPv4FQDN",
    "network_object_name": "MyFqdn",
    "network_object_value": "www.fqdn.com",
    "reason": "Object Kind: IPv4FQDN, Object Value: www.fqdn.com, Object
Name: MyFqdn, is added to Firewall: firewall_1, Group: BLOCKLIST_IN"
  },
  "inputs": {
    "cisco_asa_artifact_type": "DNS Name",
    "cisco_asa_end_range": null,
    "cisco_asa_firewall": "firewall_1",
    "cisco_asa_fqdn_ip_version": {
      "id": 427,
      "name": "IPv4FQDN"
    },
    "cisco_asa_network_object_description": "fqdn name",
    "cisco_asa_network_object_group": "BLOCKLIST_IN",
    "cisco_asa_network_object_name": "MyFqdn",
    "cisco_asa_network_object_value": "www.fqdn.com"
  },
  "metrics": {
    "execution_time_ms": 696,
    "host": "My-MacBook-Pro.local",
    "package": "fn-cisco-asa",
    "package_version": "1.0.0",
    "timestamp": "2022-06-24 12:30:41",
    "version": "1.0"
  },
  "raw": "{\"firewall\": \"firewall_1\", \"network_object_group\":
\"BLOCKLIST_IN\", \"network_object_name\": \"MyFqdn\",
\"network_object_description\": \"fqdn name\", \"network_object_kind\":
\"IPv4FQDN\", \"network_object_value\": \"www.fqdn.com\", \"reason\":
\"Object Kind: IPv4FQDN, Object Value: www.fqdn.com, Object Name: MyFqdn,
is added to Firewall: firewall_1, Group: BLOCKLIST_IN\"}",
  "reason": null,
  "success": true,
  "version": "1.0"
}
```

▶ Example Pre-Process Script:

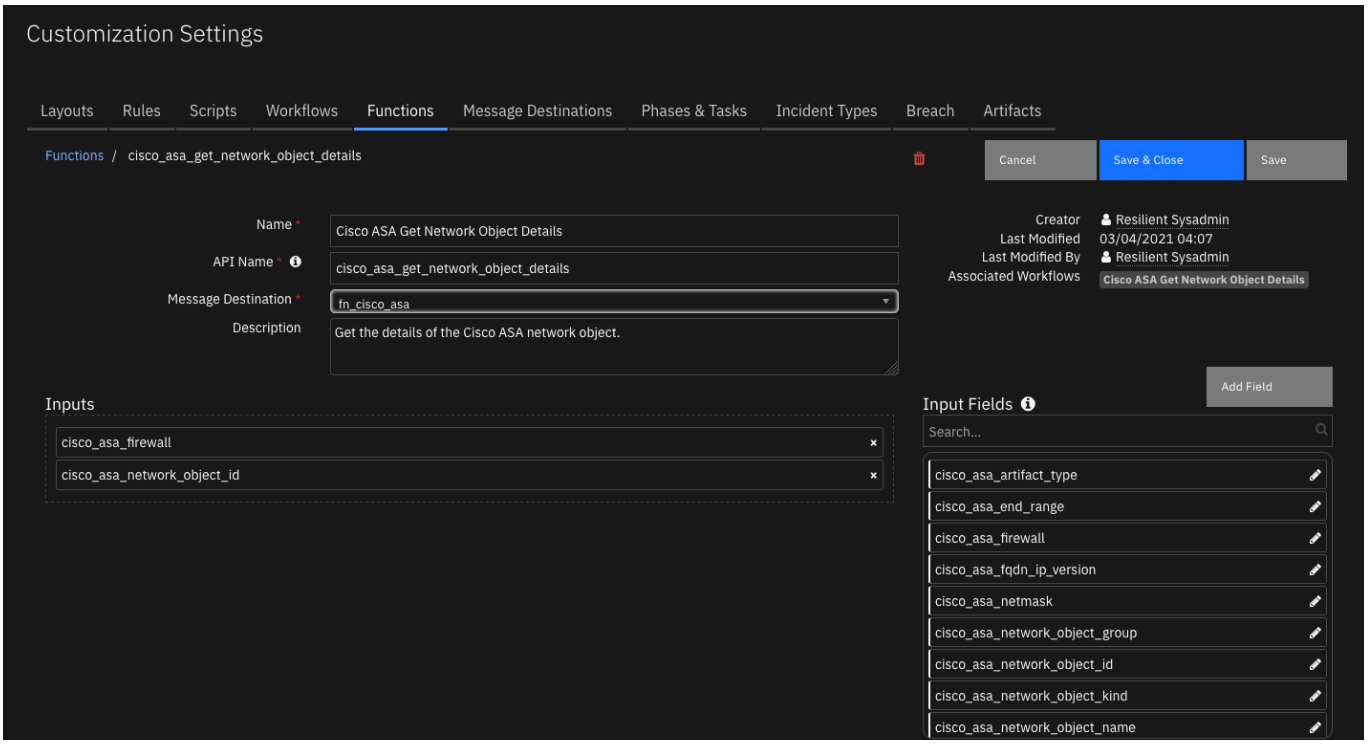
```
# Parse the firewall name and network object group from the colon
separated string
firewall_group_pair =
playbook.inputs.cisco_asa_firewall_network_object_group

# Parse the firewall group pair, which is a string in
"firewall:network_object_group" format
firewall_group_pair_list = firewall_group_pair.split(":")
inputs.cisco_asa_firewall = firewall_group_pair_list[0]
inputs.cisco_asa_network_object_group = firewall_group_pair_list[1]

# Get input from the artifact type and value
inputs.cisco_asa_network_object_value = artifact.value
inputs.cisco_asa_artifact_type = artifact.type

# Optional network object description
if playbook.inputs.cisco_asa_network_object_description:
  inputs.cisco_asa_network_object_description =
playbook.inputs.cisco_asa_network_object_description

if playbook.inputs.cisco_asa_network_object_name:
  inputs.cisco_asa_network_object_name =
playbook.inputs.cisco_asa_network_object_name
```

▶ Example Post-Process Script:

```
from datetime import datetime

results = playbook.functions.results.add_artifact_results
success = results.get("success")
content = results.get("content")
firewall = content.get("firewall")
network_object_group = content.get("network_object_group")
network_object_value = content.get("network_object_value")
network_object_name = content.get("network_object_name")
network_object_kind = content.get("network_object_kind")
if success:
  network_object_description = content.get("network_object_description")
  # Add network object as a row in the network Cisco ASA network objects
data table
  network_object_row = incident.addRow("cisco_asa_network_object_dt")
  network_object_row.cisco_asa_query_date = datetime.now()
  network_object_row.cisco_asa_firewall = firewall
  network_object_row.cisco_asa_network_object_group = network_object_group
  network_object_row.cisco_asa_network_object_kind = network_object_kind
  network_object_row.cisco_asa_network_object_value = network_object_value
  network_object_row.cisco_asa_network_object_id = network_object_name
  network_object_row.cisco_asa_network_object_description =
network_object_description
  # Update status field
  status_text = u"""<p style= "color:{color}">{status}
</p>""".format(color="green", status="Active")
  network_object_row.cisco_asa_status = helper.createRichText(status_text)
else:
  # Artifact not added to the group so add a note with the reason.
  reason = content.get("reason")
  note = u"Cisco ASA Add Artifact to Network Object Group Results:\n
Artifact value: {0}\n    Object Name: {1} \n    Object Kind: {2} was not
added to Firewall: {3}, Network Object Group: {4}\n\n{5}"
  note = note.format(network_object_value, network_object_name,
network_object_kind, firewall, network_object_group, reason)
  incident.addNote(helper.createPlainText(note))
```

Function - Cisco ASA Get Network Object Details

Get the details of the Cisco ASA network object.



Name * Cisco ASA Get Network Object Details
API Name * cisco_asa_get_network_object_details
Message Destination * fn_cisco_asa
Description Get the details of the Cisco ASA network object.

Creator Resilient Sysadmin
Last Modified 03/04/2021 04:07
Last Modified By Resilient Sysadmin
Associated Workflows Cisco ASA Get Network Object Details

Inputs
cisco_asa_firewall
cisco_asa_network_object_id

Input Fields
cisco_asa_artifact_type
cisco_asa_end_range
cisco_asa_firewall
cisco_asa_fqdn_ip_version
cisco_asa_netmask
cisco_asa_network_object_group
cisco_asa_network_object_id
cisco_asa_network_object_kind
cisco_asa_network_object_name

▶ Inputs:

Name	Type	Required	Example	Tooltip
`cisco_asa_firewall`	`text`	Yes	`-`	-
`cisco_asa_network_object_id`	`text`	No	`-`	-

▶ Outputs:

> **NOTE:** This example might be in JSON format, but `results` is a Python Dictionary on the SOAR platform.

```
results = {
  "content": {
    "description": "fqdn name",
    "host": {
      "kind": "IPv4FQDN",
      "value": "www.fqdn.com"
    },
    "kind": "object#NetworkObj",
    "name": "MyFqdn",
    "objectId": "MyFqdn",
    "selfLink": "https://192.168.1.163/api/objects/networkobjects/MyFqdn"
  },
  "inputs": {
    "cisco_asa_firewall": "firewall_1",
    "cisco_asa_network_object_id": "MyFqdn"
  },
  "metrics": {
    "execution_time_ms": 69,
    "host": "MacBook-Pro.local",
    "package": "fn-cisco-asa",
    "package_version": "1.0.0",
    "timestamp": "2022-06-24 12:30:59",
    "version": "1.0"
  },
  "raw": "{\"kind\": \"object#NetworkObj\", \"selfLink\":
\"https://192.168.1.163/api/objects/networkobjects/MyFqdn\", \"name\":
\"MyFqdn\", \"host\": {\"kind\": \"IPv4FQDN\", \"value\":
\"www.fqdn.com\"}, \"description\": \"fqdn name\", \"objectId\":
\"MyFqdn\"}",
  "reason": null,
  "success": true,
  "version": "1.0"
}
```

▶ Example Pre-Process Script:

```
inputs.cisco_asa_firewall = row.cisco_asa_firewall
inputs.cisco_asa_network_object_id = row.cisco_asa_network_object_id
```

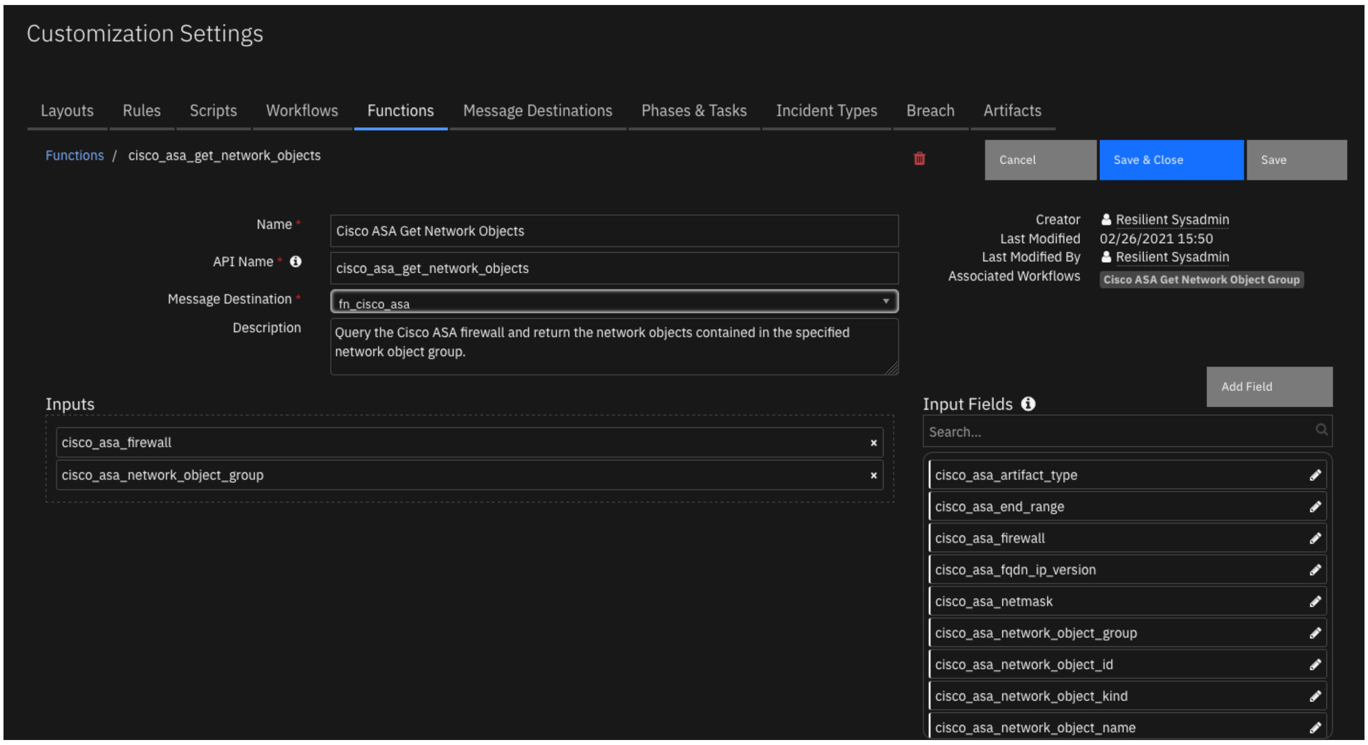
▶ Example Post-Process Script:

```
# Put the results json into a workflow property so we can call the
# convert_json_to_rich_text script to print readable formatted json in an
incident note.
results = playbook.functions.results.get_details_results
inputs = results.get("inputs")
firewall_id = inputs.get("cisco_asa_firewall")
object_id = inputs.get("cisco_asa_network_object_id")
header = u"Cisco ASA Firewall: {0} Network Object ID
```

```
{1}".format(firewall_id, object_id)

json_note = {
              "version": "1.3",
              "header": header,
              "json": results.content,
              "sort": False
            }
playbook.addProperty('convert_json_to_rich_text', json_note)
```

Function - Cisco ASA Get Network Objects

Query the Cisco ASA firewall and return the network objects contained in the specified network object group.



Name * Cisco ASA Get Network Objects
API Name * cisco_asa_get_network_objects
Message Destination * fn_cisco_asa
Description Query the Cisco ASA firewall and return the network objects contained in the specified network object group.

Creator Resilient Sysadmin
Last Modified 02/26/2021 15:50
Last Modified By Resilient Sysadmin
Associated Workflows Cisco ASA Get Network Object Group

Inputs
cisco_asa_firewall
cisco_asa_network_object_group

Input Fields
cisco_asa_artifact_type
cisco_asa_end_range
cisco_asa_firewall
cisco_asa_fqdn_ip_version
cisco_asa_netmask
cisco_asa_network_object_group
cisco_asa_network_object_id
cisco_asa_network_object_kind
cisco_asa_network_object_name

▶ Inputs:

Name	Type	Required	Example	Tooltip
`cisco_asa_firewall`	`text`	Yes	`-`	-
`cisco_asa_network_object_group`	`text`	Yes	`-`	-

▶ Outputs:

> **NOTE:** This example might be in JSON format, but `results` is a Python Dictionary on the SOAR platform.

```
results = {
  "content": {
    "member_list": [
      {
        "kind": "IPv4Address",
        "value": "192.168.10.1"
      },
      {
        "kind": "IPv4Address",
        "value": "192.168.10.2"
      },
      {
        "kind": "IPv4Address",
        "value": "192.168.10.3"
      }
    ]
  },
  "inputs": {
    "cisco_asa_firewall": "firewall_1",
    "cisco_asa_network_object_group": "BLOCKLIST_IN"
  },
  "metrics": {
    "execution_time_ms": 551,
    "host": "MacBook-Pro.local",
    "package": "fn-cisco-asa",
    "package_version": "1.0.0",
    "timestamp": "2022-06-24 12:23:40",
    "version": "1.0"
  },
  "raw": "{\"member_list\": [{\"kind\": \"IPv4Address\", \"value\":
\"192.168.10.1\"}, {\"kind\": \"IPv4Address\", \"value\":
\"192.168.10.2\"}, {\"kind\": \"IPv4Address\", \"value\":
\"192.168.10.3\"}]}",
  "reason": null,
  "success": true,
  "version": "1.0"
}
```

▶ Example Pre-Process Script:

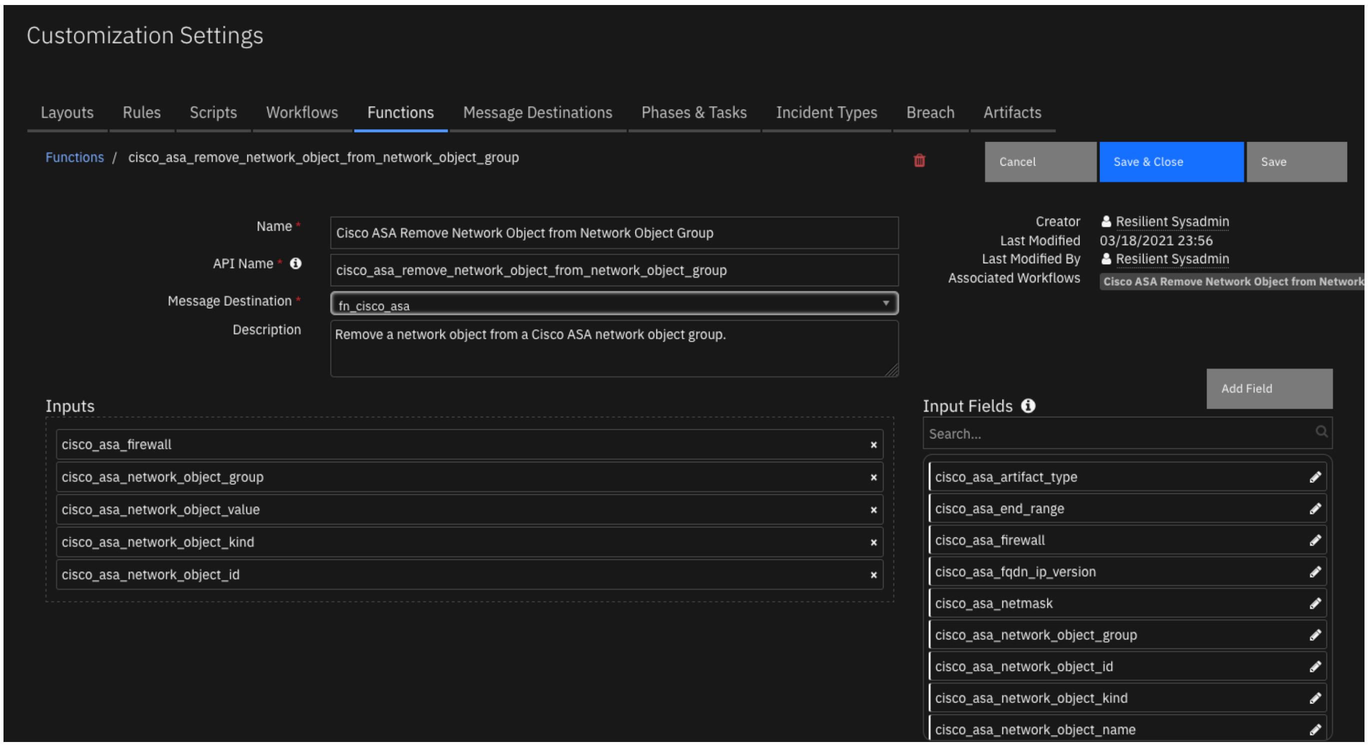
```
# Get the firewall network object group pair
firewall_group_pair =
playbook.inputs.cisco_asa_firewall_network_object_group

# Parse the firewall group pair, which is a string in
"firewall:network_object_group" format
firewall_group_pair_list = firewall_group_pair.split(":")
inputs.cisco_asa_firewall = firewall_group_pair_list[0]
inputs.cisco_asa_network_object_group = firewall_group_pair_list[1]
```

▶ Example Post-Process Script:

```
from datetime import datetime

results = playbook.functions.results.get_network_objects_results
content = results.get("content", {})
member_list = content.get("member_list", [])
firewall = results.inputs.get("cisco_asa_firewall")
network_object_group =
results.inputs.get("cisco_asa_network_object_group")

# Add each email as a row in the query results data table
for network_object in member_list:
  network_object_row = incident.addRow("cisco_asa_network_object_dt")
  network_object_row.cisco_asa_query_date = datetime.now()
  network_object_row.cisco_asa_firewall = firewall
  network_object_row.cisco_asa_network_object_group = network_object_group

  if network_object.get("kind")  == 'object#NetworkObj':
    network_object_row.cisco_asa_network_object_id =
network_object.get("objectId")
    network_object_row.cisco_asa_network_object_description =
network_object.get("description")
    host = network_object.get("host")
    network_object_row.cisco_asa_network_object_kind = host.get("kind")
    network_object_row.cisco_asa_network_object_value = host.get("value")
  else:
    network_object_row.cisco_asa_network_object_kind =
network_object.get("kind")
    network_object_row.cisco_asa_network_object_value =
network_object.get("value")

  status_text = u"""<p style= "color:{color}">{status}
</p>""".format(color="green", status="Active")
  network_object_row.cisco_asa_status = helper.createRichText(status_text)
```

Function - Cisco ASA Remove Network Object from Network Object Group

Remove a network object from a Cisco ASA network object group.



Functions / cisco_asa_remove_network_object_from_network_object_group

Name * Cisco ASA Remove Network Object from Network Object Group
API Name * cisco_asa_remove_network_object_from_network_object_group
Message Destination * fn_cisco_asa
Description Remove a network object from a Cisco ASA network object group.

Creator Resilient Sysadmin
Last Modified 03/18/2021 23:56
Last Modified By Resilient Sysadmin
Associated Workflows Cisco ASA Remove Network Object from Network

Inputs
- cisco_asa_firewall
- cisco_asa_network_object_group
- cisco_asa_network_object_value
- cisco_asa_network_object_kind
- cisco_asa_network_object_id

Input Fields
- cisco_asa_artifact_type
- cisco_asa_end_range
- cisco_asa_firewall
- cisco_asa_fqdn_ip_version
- cisco_asa_netmask
- cisco_asa_network_object_group
- cisco_asa_network_object_id
- cisco_asa_network_object_kind
- cisco_asa_network_object_name

▶ Inputs:

Name	Type	Required	Example	Tooltip
`cisco_asa_firewall`	`text`	Yes	`-`	-
`cisco_asa_network_object_group`	`text`	Yes	`-`	-
`cisco_asa_network_object_id`	`text`	No	`-`	-
`cisco_asa_network_object_kind`	`text`	Yes	`-`	-
`cisco_asa_network_object_value`	`text`	Yes	`-`	-

▶ Outputs:

> **NOTE:** This example might be in JSON format, but `results` is a Python Dictionary on the SOAR platform.

```
results = {
  "content": "Object Kind: IPv4Address, Object Value: 192.168.10.1, Object Name: None, is removed from Firewall: firewall_1, Group: BLOCKLIST_IN",
  "inputs": {
    "cisco_asa_firewall": "firewall_1",
    "cisco_asa_network_object_group": "BLOCKLIST_IN",
    "cisco_asa_network_object_id": null,
    "cisco_asa_network_object_kind": "IPv4Address",
    "cisco_asa_network_object_value": "192.168.10.1"
  },
  "metrics": {
    "execution_time_ms": 512,
    "host": "MacBook-Pro.local",
    "package": "fn-cisco-asa",
    "package_version": "1.0.0",
    "timestamp": "2022-06-24 12:24:30",
    "version": "1.0"
  },
  "raw": "\"Object Kind: IPv4Address, Object Value: 192.168.10.1, Object
Name: None, is removed from Firewall: firewall_1, Group: BLOCKLIST_IN\"",
  "reason": null,
  "success": true,
  "version": "1.0"
}
```

▶ Example Pre-Process Script:

```
inputs.cisco_asa_firewall = row.cisco_asa_firewall
inputs.cisco_asa_network_object_group = row.cisco_asa_network_object_group
inputs.cisco_asa_network_object_kind = row.cisco_asa_network_object_kind
inputs.cisco_asa_network_object_value = row.cisco_asa_network_object_value
inputs.cisco_asa_network_object_id = row.cisco_asa_network_object_id
```

▶ Example Post-Process Script:

```
from datetime import datetime

results = playbook.functions.results.remove_results

if results.success:
  text = "Removed"
else:
  text = "NotFound"
  note = "Remove Network Object From Network Object Group Results:\n\n
{0}".format(results.content)
  incident.addNote(helper.createPlainText(note))

status_text = u"""<p style= "color:{color}">{status}
</p>""".format(color="red", status=text)
row['cisco_asa_status'] = helper.createRichText(status_text)
row["cisco_asa_query_date"] = datetime.now()
```

Script - Convert JSON to rich text v1.3

This script converts a json object into a hierarchical display of rich text and adds the rich text to an incident's rich text (custom) field or an incident note. A workflow property is used to share the json to convert and identify parameters used on how to perform the conversion. Typically, a function will create workflow property and this script will run after that function to perform the conversion.

Features:

- Display the hierarchical nature of json, presenting the json keys (sorted if specified) as bold labels

- Provide links to found URLs
- Create either an incident note or add results to an incident (custom) rich text field.

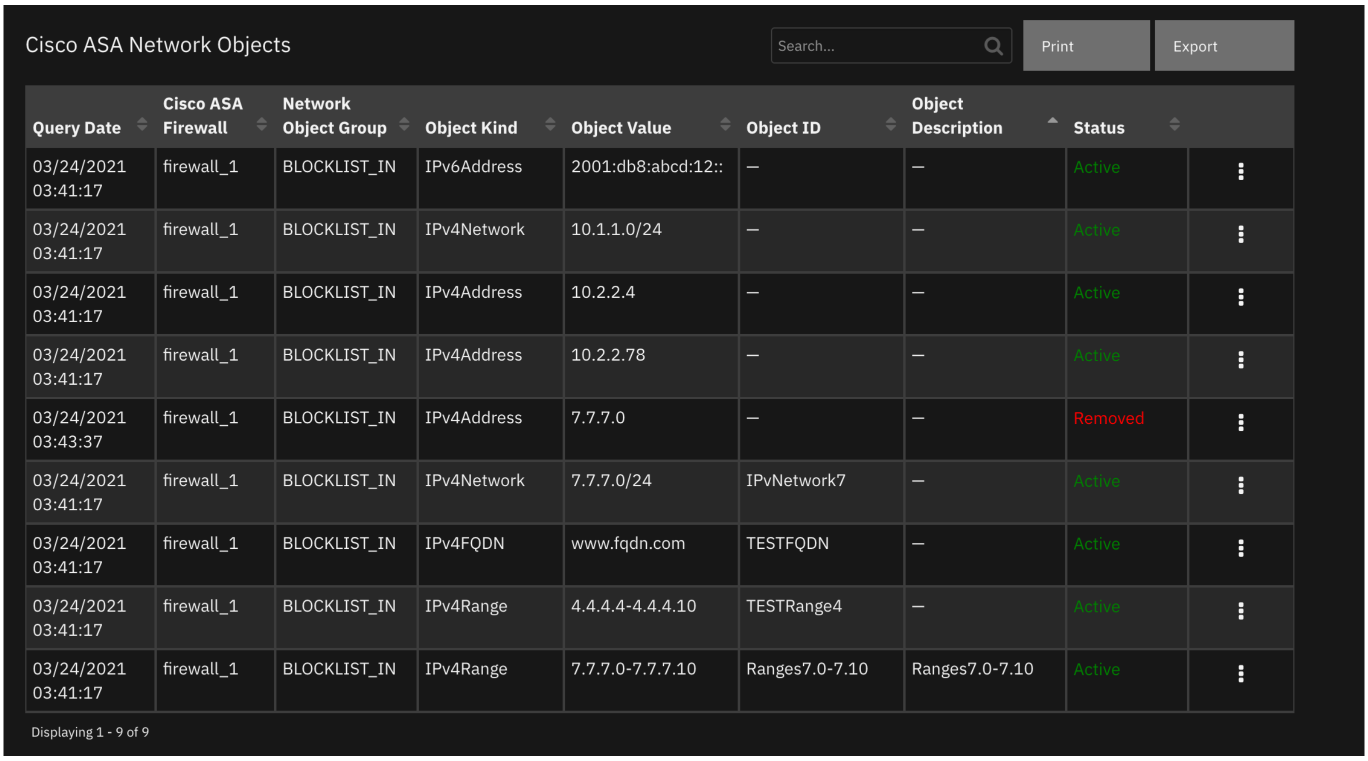
Object: incident

▶ Script Text:

```
# (c) Copyright IBM Corp. 2010, 2022. All Rights Reserved.
VERSION = 1.3
"""
  This script converts a json object into a hierarchical display of rich
text and adds the rich text to an incident's rich text (custom) field or
an incident note.
  A workflow property is used to define the json to convert and identify
parameters used on how to perform the conversion.
  Typically, a function will create workflow property and this script will
run after that function to perform the conversion.
  Features:
    * Display the hierarchical nature of json, presenting the json keys as
bold labels
    * Provide links to found URLs
    * Create either an incident note or add results to an incident
(custom) rich text field.

  In order to use this script, define a workflow property called:
convert_json_to_rich_text, to define the json and parameters to use for
the conversion.
  Workflow properties can be added using a command similar to this:
  workflow.addProperty('convert_json_to_rich_text', {
    "version": 1.3,
    "header": "Artifact scan results for: {}".format(artifact.value),
    "padding": 10,
    "separator": u"<br />",
    "sort": True,
    "json": results.content,
    "json_omit_list": ["omit"],
    "incident_field": None
  })

  Format of workflow.property.convert_json_to_rich_text:
  {
    "version": 1.3, [this is for future compatibility]
    "header": str, [header line to add to converted json produced or None.
Ex: Results from scanning artifact: xxx. The header may contain rich text
tags]
    "padding": 10, [padding for nested json elements, or defaults to 10]
    "separator": u"<br />"|list such as ['<span>','</span>'], [html
separator between json keys and lists or defaults to html break: '<br />'.
                                                If a list, then the data
is brackets by the pair specified]
    "sort": True|False, [sort the json keys at each level when displayed]
    "json": json, [required json to convert]
    "json_omit_list": [list of json keys to exclude or None]
    "incident_field": "<incident_field>" [indicates a builtin rich text
incident field, such as 'description'
                                          or a custom rich text field in
the format: 'properties.<field>'. default: create an incident note]
  }

  For playbooks, use playbook.addProperty() with the same format as
workflow.addProperty()

  Playbooks can also use
playbook.functions.results.convert_json_to_rich_text using the standard
function output which contains the 'content' json element.
  When using playbook.functions.results.convert_json_to_rich_text with
standard function results, all the defaults for padding, separator, etc.
are used.
"""

import re

# needed for python 3
try:
    unicode("abc") # fails in py3
    py2 = True
except:
    unicode = str
    py2 = False


rc = re.compile(r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+#\?]|[!*\(\),]|(?:%
[0-9a-fA-F][0-9a-fA-F]))+')

class ConvertJson:
    """Class to hold the conversion parameters and perform the
conversion"""

    def __init__(self, omit_keys=[], padding=10, separator=u"<br />",
sort_keys=False):
        self.omit_keys = omit_keys
        self.padding = padding
        self.separator = separator
        self.sort_keys = sort_keys


    def format_link(self, item):
        """[summary]
          Find embedded urls (http(s)) and add html anchor tags to display
as links
          Args:
              item ([string])

          Returns:
              [str]: None|original text if no links|text with html links
        """
        formatted_item = item
        if py2:
            num_type = bool(item and isinstance(item, (int, long, bool,
float)))
        else:
            num_type = bool(item and isinstance(item, (int, bool, float)))

        if item and not num_type:
            list = rc.findall(item)
            if list:
                for link in list:
                    formatted_item = formatted_item.replace(link, u"<a
target='blank' href='{0}'>{0}</a>".format(link))

        return formatted_item

    def expand_list(self, list_value, is_list=False):
        """[summary]
          convert items to html, adding indents to nested dictionaries.
          Args:
              list_value ([dict|list]): json element

          Returns:
              [str]: html converted code
        """
        if not isinstance(list_value, list):
            return self.format_link(list_value)
        elif not list_value:
            return u"None<br>"

        try:
            items_list = []  # this will ensure list starts on second line
of key label
            for item in list_value:
                if isinstance(item, dict):
                    result = self.convert_json_to_rich_text(item)
                    if is_list:
                        items_list.append(u"<li>{}</li>".format(result))
                    else:
                        items_list.append(result)
                elif isinstance(item, list):
                    items_list.append(self.expand_list(item,
is_list=True))
                elif is_list:
                    items_list.append(u"<li>{}
</li>".format(self.format_link(unicode(item))))
                else:
                    items_list.append(self.format_link(unicode(item)))

            expand_list_result = self.add_separator(self.separator if not
is_list else u"",
                                                    items_list,
                                                    is_list=is_list)

            if is_list:
                return u"<ul>{}</ul>".format(expand_list_result)
            else:
                return u"<div style='padding:5px'>{}
</div>".format(expand_list_result)
        except Exception as err:
            return str(err)

    def convert_json_to_rich_text(self, sub_dict):
        """[summary]
          Walk dictionary tree and convert to html for better display
          Args:
              sub_dict ([type]): [description]

          Returns:
              [type]: [description]
        """
        notes = []
        if sub_dict and isinstance(sub_dict, (list, dict)):
            if isinstance(sub_dict, list):
                expanded_list = self.expand_list(sub_dict, is_list=True)
                notes.append(self.add_separator(self.separator,
expanded_list))
            else:
                keys = sorted (sub_dict.keys()) if self.sort_keys else
sub_dict.keys()

                for key in keys:
                    if key not in self.omit_keys:
                        value = sub_dict[key]
                        is_list = isinstance(value, list)
                        item_list = [u"<strong>{0}</strong>:
".format(key)]
                        if isinstance(value, dict):
                            convert_result =
self.convert_json_to_rich_text(value)
                            if convert_result:
                                item_list.append(u"<div style='padding:
{}px'>{}</div>".format(self.padding, convert_result))
                            else:
                                item_list.append(u"None<br>")
                        else:
                            item_list.append(self.expand_list(value,
is_list=is_list))

                        notes.append(self.add_separator(self.separator,
u"".join(make_unicode(v) for v in item_list), is_list=is_list))

        result_notes = u"".join(notes)
        if isinstance(self.separator, list):
            return result_notes
        else:
            return result_notes.replace(
                u"</div>{0}".format(self.separator), u"</div>").replace(
                u"{0}</div>".format(self.separator), u"</div>"
            )  # tighten up result

    def add_separator(self, separator, items, is_list=False):
        """
        apply the separator to the data
        :param separator: None, str or list such as ['<span>', '</span>']
        :param items: str or list to add separator
        :return: text with separator applied
        """
        _items = items

        if not _items:
            return "<br>"

        if not isinstance(_items, list):
            _items = [_items]

        if isinstance(separator, list):
            return u"".join([u"{}{}{}".format(separator[0], item,
separator[1]) for item in _items])

        return u"{}{}".format(separator.join(_items), separator if not
is_list else u"")

def make_unicode(value):
    if value is None:
        return 'None'

    return unicode(value)

def get_results(property_name):
    if playbook and playbook.functions.results[property_name] is not None:
        return playbook.functions.results[property_name]
    elif playbook and playbook.properties[property_name] is not None:
        return playbook.properties[property_name]
    elif workflow and workflow.properties[property_name] is not None:
        return workflow.properties[property_name]

    return None

def get_properties(property_name):
    """
    Logic to collect the json and parameters from a workflow property.
    Args:
      property_name: workflow property to reference
    Returns:
      padding, separator, header, json_omit_list, incident_field, json,
sort_keys
    """
    result_properties = get_results(property_name)
    if not result_properties:
        helper.fail("Playbook/workflow property not found:
{}".format(property_name))
```

```
    padding = int(result_properties.get("padding", 10))
    separator = result_properties.get("separator", u"<br />")
    if isinstance(separator, list) and len(separator) != 2:
        helper.fail("list of separators should be specified as a pair such
as ['<div>', '</div>']: {}".format(separator))

    header = result_properties.get("header")
    sort_keys = bool(result_properties.get("sort", False))
    json_omit_list = result_properties.get("json_omit_list")
    if not json_omit_list:
        json_omit_list = []
    incident_field = result_properties.get("incident_field")

    # workflow formatted content is 'json'. Standard functions is
'content'
    json = result_properties.get("json") if result_properties.get("json")
else result_properties.get("content")
    json_err = None
    # is there an issue we need handle now?
    if not json and \
        result_properties.get("success") == False and
result_properties.get("reason"):
        json_err = result_properties.get("reason")

    return padding, separator, header, json_omit_list, incident_field,
json, json_err, sort_keys


## S T A R T
padding, separator, header, json_omit_list, incident_field, json,
json_err, sort_keys = get_properties('convert_json_to_rich_text')
if json_err:
    result = "Result error: {}".format(json_err)
else:
    if header:
        if isinstance(separator, list):
            hdr = u"{0}{1}{2}".format(separator[0], header, separator[1])
        else:
            hdr = u"{0}{1}".format(header, separator)
    else:
        hdr = u""

    convert = ConvertJson(omit_keys=json_omit_list, padding=padding,
separator=separator, sort_keys=sort_keys)
    converted_json = convert.convert_json_to_rich_text(json)
    result = u"{}{}".format(hdr, converted_json if converted_json else
"\nNone")

rich_text_note = helper.createRichText(result)
if incident_field:
    incident[incident_field] = rich_text_note
else:
    incident.addNote(rich_text_note)
```

Data Table - Cisco ASA Network Objects


Cisco ASA Network Objects

Query Date	Cisco ASA Firewall	Network Object Group	Object Kind	Object Value	Object ID	Object Description	Status	
03/24/2021 03:41:17	firewall_1	BLOCKLIST_IN	IPv6Address	2001:db8:abcd:12::	–	–	Active	⋮
03/24/2021 03:41:17	firewall_1	BLOCKLIST_IN	IPv4Network	10.1.1.0/24	–	–	Active	⋮
03/24/2021 03:41:17	firewall_1	BLOCKLIST_IN	IPv4Address	10.2.2.4	–	–	Active	⋮
03/24/2021 03:41:17	firewall_1	BLOCKLIST_IN	IPv4Address	10.2.2.78	–	–	Active	⋮
03/24/2021 03:43:37	firewall_1	BLOCKLIST_IN	IPv4Address	7.7.7.0	–	–	Removed	⋮
03/24/2021 03:41:17	firewall_1	BLOCKLIST_IN	IPv4Network	7.7.7.0/24	IPvNetwork7	–	Active	⋮
03/24/2021 03:41:17	firewall_1	BLOCKLIST_IN	IPv4FQDN	www.fqdn.com	TESTFQDN	–	Active	⋮
03/24/2021 03:41:17	firewall_1	BLOCKLIST_IN	IPv4Range	4.4.4.4-4.4.4.10	TESTRange4	–	Active	⋮
03/24/2021 03:41:17	firewall_1	BLOCKLIST_IN	IPv4Range	7.7.7.0-7.7.7.10	Ranges7.0-7.10	Ranges7.0-7.10	Active	⋮

Displaying 1 - 9 of 9

API Name:

cisco_asa_network_object_dt

Columns:

Column Name	API Access Name	Type	Tooltip
Cisco ASA Firewall	`cisco_asa_firewall`	`text`	-
Network Object Group	`cisco_asa_network_object_group`	`text`	-
Object Description	`cisco_asa_network_object_description`	`text`	-
Object ID	`cisco_asa_network_object_id`	`text`	-
Object Kind	`cisco_asa_network_object_kind`	`text`	-
Object Value	`cisco_asa_network_object_value`	`text`	-
Query Date	`cisco_asa_query_date`	`datetimepicker`	-
Status	`cisco_asa_status`	`textarea`	-

Playbooks

Playbook Name	Description	Object	Status
Cisco ASA: Add FQDN to Network Object Group (PB)	Add an DNS artifact to a Cisco ASA network object group. If the artifact is not added to the group an error message is written to an incident note.	artifact	`enabled`
Cisco ASA: Add IP Address to Network Object Group (PB)	Add IP address artifact to Cisco ASA network object group. If the artifact is not added to the group an error message is written to an incident note.	artifact	`enabled`
Cisco ASA: Add IP Range to Network Object Group (PB)	Add IP range to a Cisco ASA network object group. If there is an error write it to an incident note.	artifact	`enabled`
Cisco ASA: Add IPv4Network to Network Object Group (PB)	Add an IPv4 network to a Cisco ASA network object group. If there is an error write the error to an incident note.	artifact	`enabled`
Cisco ASA: Add IPv6Network to Network Object Group (PB)	Add in IPv6 IP network to a Cisco ASA network object group. If an error occurs write it to an incident note.	artifact	`enabled`
Cisco ASA: Get Network Object Details (PB)	None	cisco_asa_network_object_dt	`enabled`
Cisco ASA Get Network Object Group	Get the network objects of the specified network object group and write them to the Cisco ASA Network Objects data table.	incident	`enabled`
Cisco ASA: Remove Network Object from Network Object Group (PB)	Remove the network object from the specified Cisco ASA network object group. The status column of the row is marked as Removed in the data table.	cisco_asa_network_object_dt	`enabled`

Troubleshooting & Support

Refer to the documentation listed in the Requirements section for troubleshooting information.

For Support

This is a IBM Community provided App. Please search the Community ibm.biz/soarcommunity for assistance.