IBM Resilient

resilient

Incident Response Platform Integrations

Slack Function V1.0.0

Release Date: November 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 Slack Integration.

Overview

Slack is an online communications solution allowing communities to communicate as groups or directly with each other through conversations and video conferences in Slack channels. This Resilient platform functions-based integration allows Incident, Note, Artifact, Task and Attachment data to be shared in Slack, with an optional custom text message. User can use the activity fields to create private or public channels in Slack workspace and invite Slack users to the channel with their emails. User can also customize what Incident, Note, Task and Artifact data they wish to post in a channel. A function to export conversation history from the chosen Slack channel to a text file, save the text file as an Incident or Task Attachment and archive the Slack channel is available as well. This integration can be used to extend the communication about Resilient data to additional groups and individuals.

Eight example workflows are present which Incidents, Notes, Artifacts, Tasks and Attachments data to be shared and Slack channel archived.

The remainder of this document describes the included functions, how to configure them in custom workflows, and any additional customization options.

Installation

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

- Resilient platform is version 31 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.)

Slack configuration

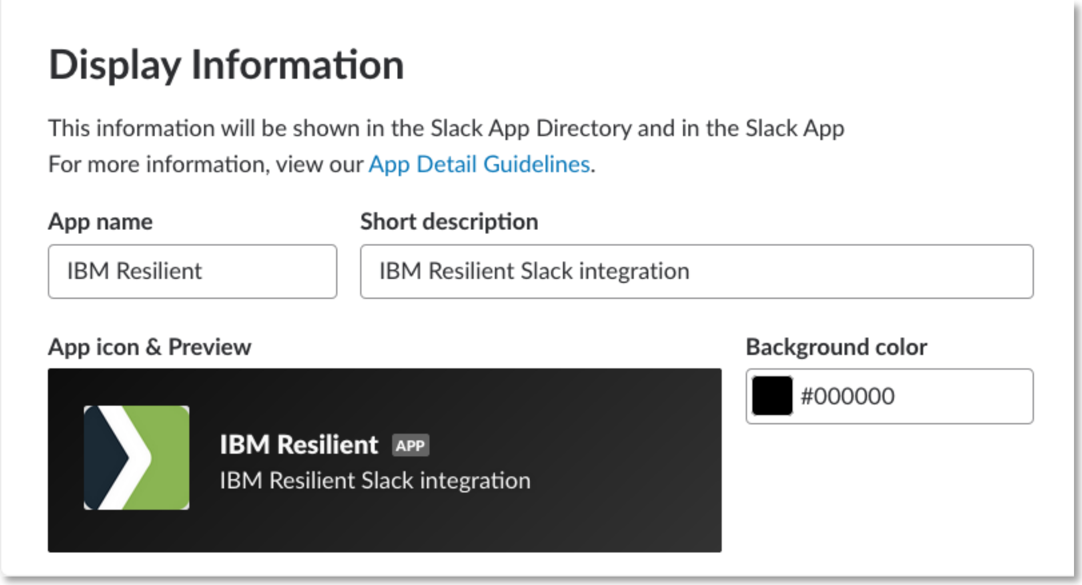
Prior to installing Slack function, follow the Slack documentation (https://api.slack.com/slack-apps) on building a new Slack App.

In the Basic Information find Display Information section and configure your App name and upload the App icon. Your App's name will be used for message authorship.

How message authorship is attributed varies by a few factors, with some behaviors varying depending on the kinds of tokens you're using to post a message.

The App's name and icon will be used as the author of the posted messages. User can change the message authorship in the app.config file or the example workflows.

When uploading files the name of the authenticated user of the Slack App will be used for authorship. This behavior can be changed by adding a Bot User.



Display Information

This information will be shown in the Slack App Directory and in the Slack App
For more information, view our App Detail Guidelines.

App name: IBM Resilient

Short description: IBM Resilient Slack integration

App icon & Preview

Background color: #000000

Our integration will need certain Permissions enabled. After you create a Slack App and set it's Display Information, select Permissions in the Add features and functionality section.

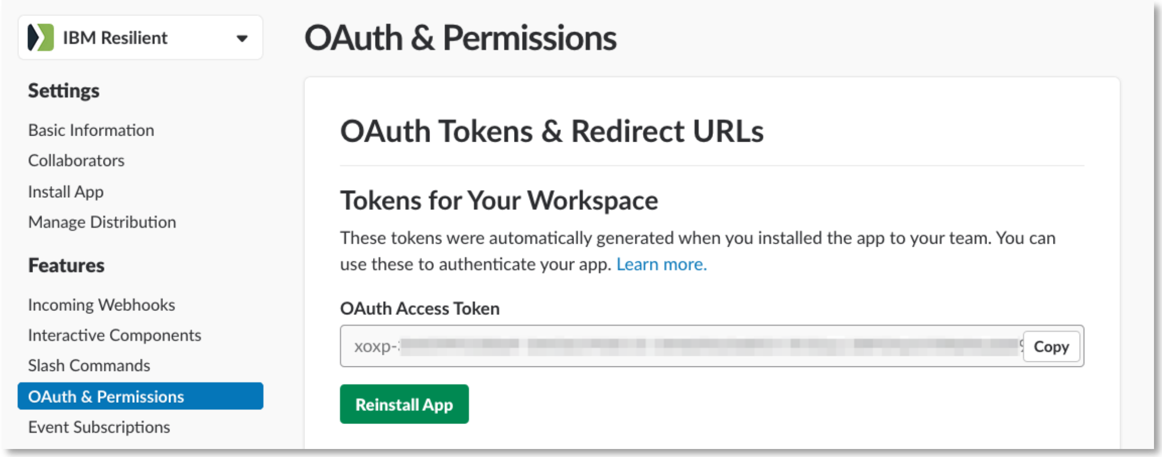
Scroll down to Scope section and add following permission scopes:

- chat:write:bot to send messages as Slack App name

- chat:write:user to send messages as user
- channels:read to access information about user's public channels
- groups:read to access information about user's private channels

  Note: The function will only retrieve those private channels from your Slack workspace in which the Slack App's authorized user has been invited to.
- channels:write to modify your public channels
- groups:write to modify your private channels
- users:read.email to view email addresses of people on this workspace
- channels:history to access user's public channels
- groups:history to access content in user's private channels
- users:read access your workspace's profile information
- files:write:user upload and modify files as user

Make sure to save changes and then click on Install App to Workspace in the OAuth Tokens & Redirect URLs section. Installing App to your Workspace will generate the OAuth Access Token needed for the integration.



OAuth & Permissions

OAuth Tokens & Redirect URLs

Tokens for Your Workspace

These tokens were automatically generated when you installed the app to your team. You can use these to authenticate your app. Learn more.

OAuth Access Token

xoxp-[redacted]

Install the Python components

The slack package contains Python components that will be 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.

Ensure that the environment is up to date,

```
sudo pip install --upgrade pip
sudo pip install --upgrade setuptools
sudo pip install --upgrade resilient-circuits
```

To install the package:

```
sudo pip install --upgrade fn_slack-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.

Perform the following to configure and run the integration:

1. Using sudo, become the integration user.

   ```
   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.

   a. In the [resilient] section, ensure that you provide all the information needed to connect to the Resilient platform.

   b. In the [fn_slack] section, edit the settings as follows:

   ```
   # Slack app OAuth Access Token
   api_token=xoxp-xxxxxxxxx-xxxxxxxxxxxx-xxxxxxxxxxxxx-xxxxxxxxxxx

   # Username represents the default submission author.
   # Used together with 'as_user=False'.
   # You can also update the username on the Workflow.
   username=Resilient
   ```

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.

Configuration of resilient-circuits for restartability

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:

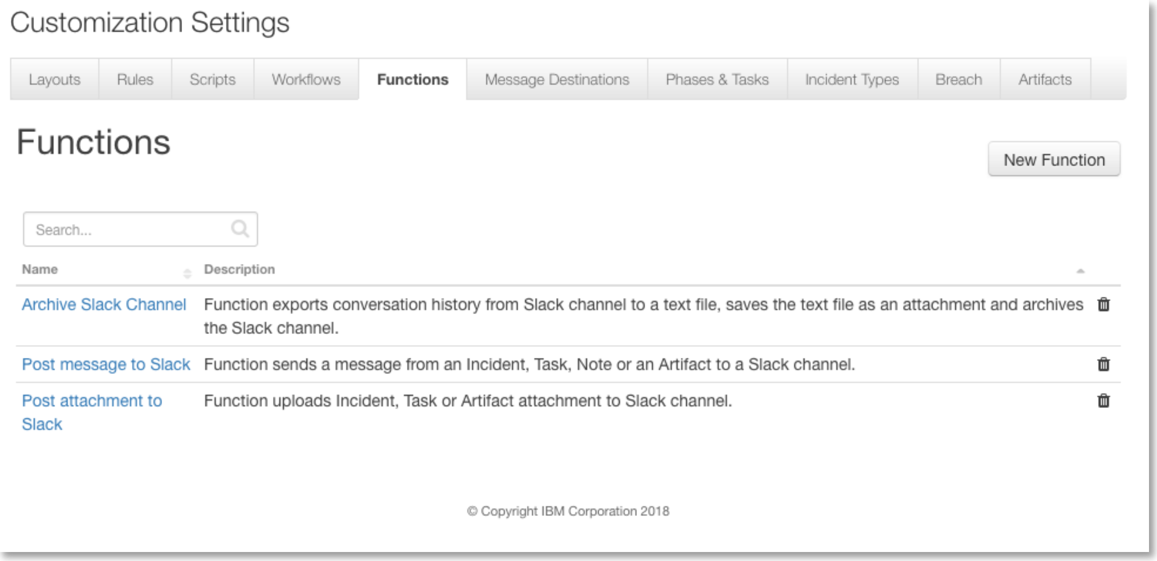
   ```
   sudo systemctl resilient_circuits [start|stop|restart|status]
   ```

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"
```

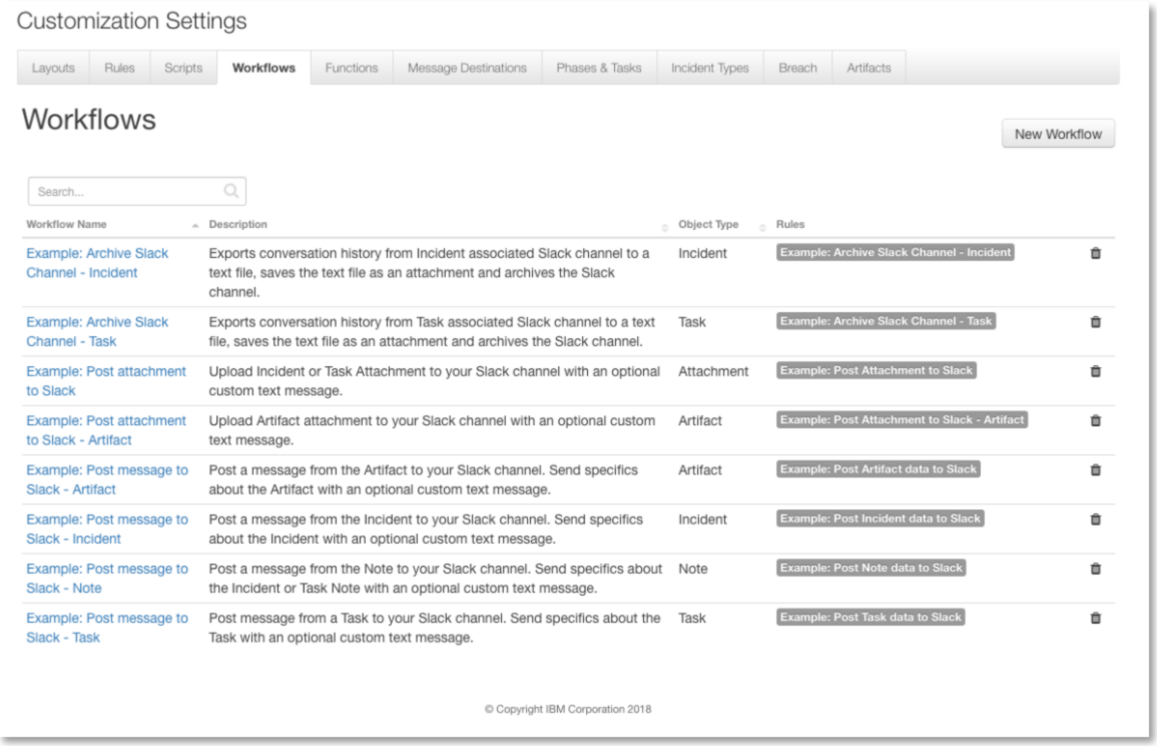
Function Descriptions

Once the function package deploys, you can view three functions in the Resilient platform Functions tab, as shown below.



Name	Description
Archive Slack Channel	Function exports conversation history from Slack channel to a text file, saves the text file as an attachment and archives the Slack channel.
Post message to Slack	Function sends a message from an Incident, Task, Note or an Artifact to a Slack channel.
Post attachment to Slack	Function uploads Incident, Task or Artifact attachment to Slack channel.

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



Workflow Name	Description	Object Type	Rules
Example: Archive Slack Channel - Incident	Exports conversation history from Incident associated Slack channel to a text file, saves the text file as an attachment and archives the Slack channel.	Incident	Example: Archive Slack Channel - Incident
Example: Archive Slack Channel - Task	Exports conversation history from Task associated Slack channel to a text file, saves the text file as an attachment and archives the Slack channel.	Task	Example: Archive Slack Channel - Task
Example: Post attachment to Slack	Upload Incident or Task Attachment to your Slack channel with an optional custom text message.	Attachment	Example: Post Attachment to Slack
Example: Post attachment to Slack - Artifact	Upload Artifact attachment to your Slack channel with an optional custom text message.	Artifact	Example: Post Attachment to Slack - Artifact
Example: Post message to Slack - Artifact	Post a message from the Artifact to your Slack channel. Send specifics about the Artifact with an optional custom text message.	Artifact	Example: Post Artifact data to Slack
Example: Post message to Slack - Incident	Post a message from the Incident to your Slack channel. Send specifics about the Incident with an optional custom text message.	Incident	Example: Post Incident data to Slack
Example: Post message to Slack - Note	Post a message from the Note to your Slack channel. Send specifics about the Incident or Task Note with an optional custom text message.	Note	Example: Post Note data to Slack
Example: Post message to Slack - Task	Post message from a Task to your Slack channel. Send specifics about the Task with an optional custom text message.	Task	Example: Post Task data to Slack

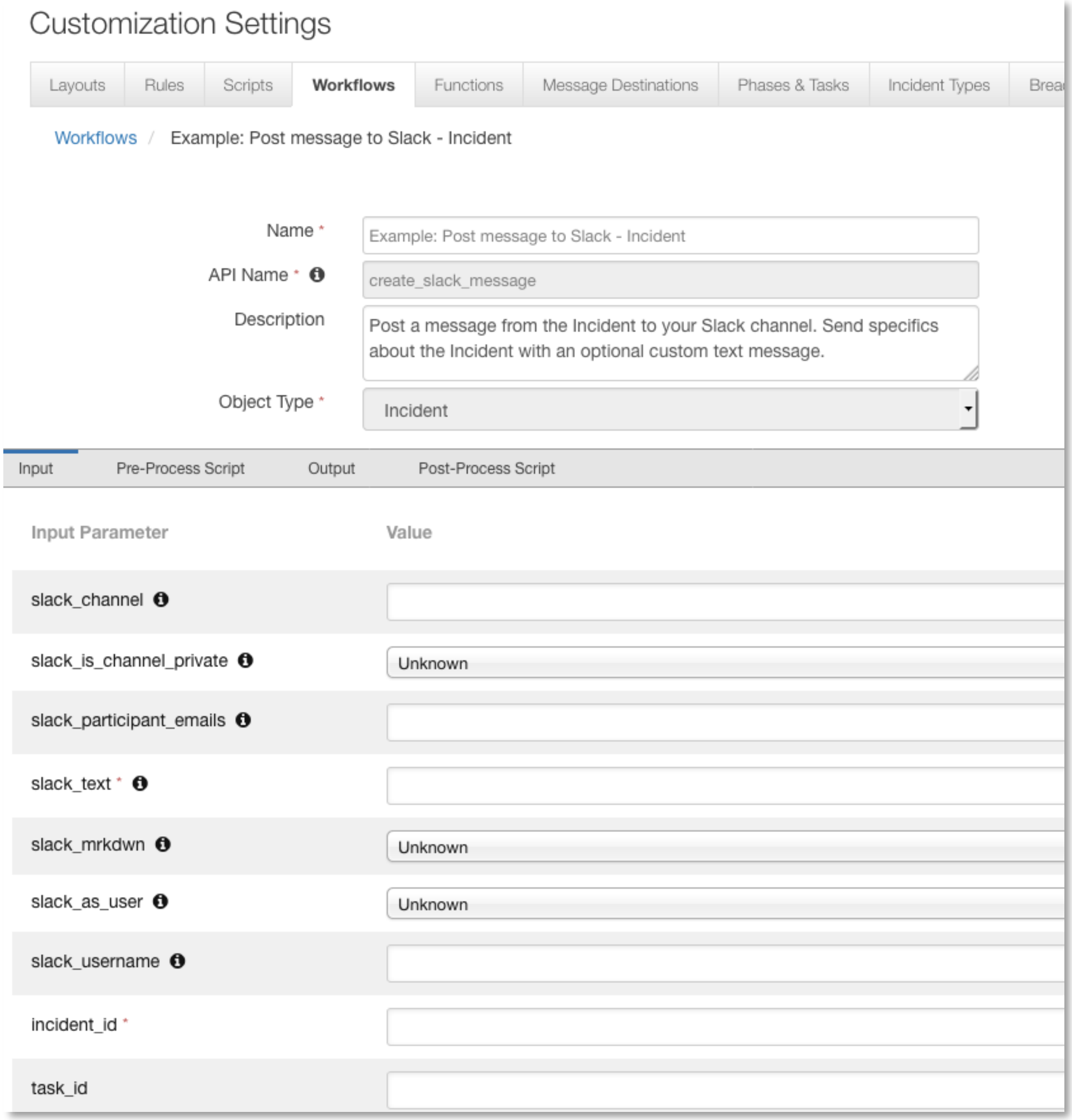
Fn_slack: slack_post_message

Function that sends a message from an Incident, Task, Note or an Artifact to a Slack channel.

This function has four example workflows that demonstrate posting a message from the Incident, Task, Note and Artifact to a chosen Slack channel. Example workflows show how to customize what data from Incident, Note, Task and Artifact to post in Slack and enable you to add an optional custom text message with it.

You can see the input fields Slack function takes in the Example: Post message to Slack – Incident, Example: Post message to Slack – Task, Example: Post message to Slack – Note and Example: Post message to Slack – Artifact workflows.

The function input fields start with `slack_` can be set on the Input Tab of the Workflow:



Name *	Example: Post message to Slack - Incident
API Name *	create_slack_message
Description	Post a message from the Incident to your Slack channel. Send specifics about the Incident with an optional custom text message.
Object Type *	Incident

Input Parameter	Value
slack_channel	
slack_is_channel_private	Unknown
slack_participant_emails	
slack_text *	
slack_mrkdwn	Unknown
slack_as_user	Unknown
slack_username	
incident_id *	
task_id	

- slack_channel is name of the existing or a new slack channel used to send message to
- slack_is_channel_private indicates if the channel you are posting to should be private

- slack_participant_emails comma separated list of emails belonging to Slack users in your workspace that will be added to your channel
- slack_text can be a text message or a container field to retain JSON fields to send to Slack
- slack_mrkdwn disable Slack markup parsing by setting to false
- slack_as_user if set to true, the authenticated user of the Slack App will appear as the author of the message, ignoring any values provided for slack_username
- slack_username replaces your Slack App's name to appear as the author of the message, must be used in conjunction with slack_as_user set to false, otherwise ignored

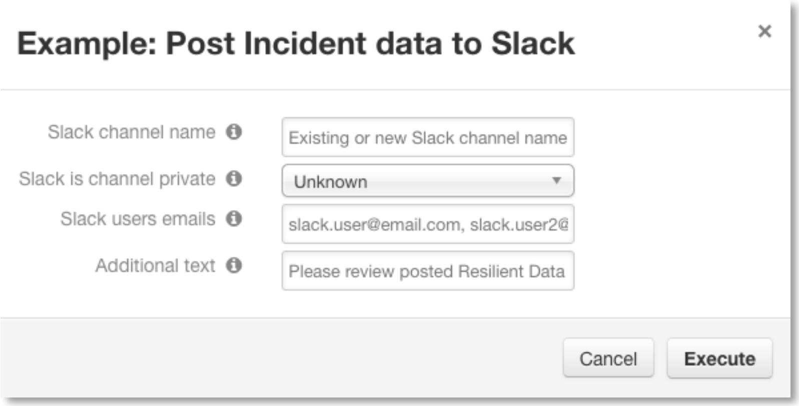
The default settings for posting messages in the function are:

- parse="none" in order for Slack not to perform any processing on the message, it will keep all markup formatting
- link_names=1 in order for Slack to linkify URLs, channel names (starting with a '#') and username ids (starting with an '<@ user_id >') We can send messages like "Hey user <@UCNC5K34J> check out #random" to Slack and it will find and link channel names and usernames.

Refer to the Slack API chat.postMessage documentation on more detailed explanation of the Slack arguments.

Inputs

Some of the function input fields can also be set when clicking a Menu Item:



Example: Post Incident data to Slack

Slack channel name: Existing or new Slack channel name

Slack is channel private: Unknown

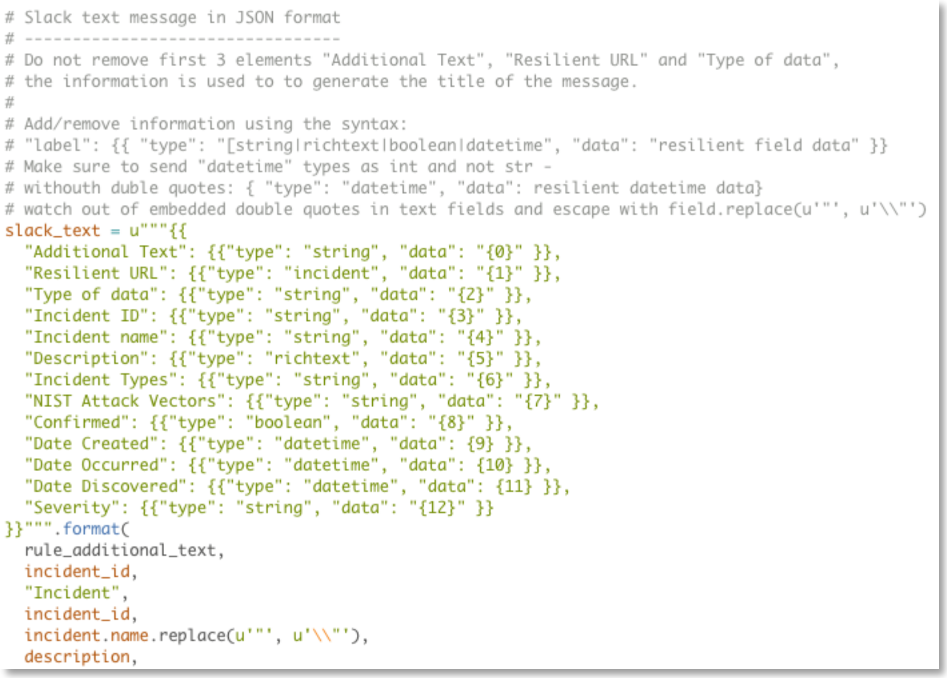
Slack users emails: slack.user@email.com, slack.user2@

Additional text: Please review posted Resilient Data

- Slack channel name is a name of the existing Slack Workspace channel or a new Slack channel you are posting to, channel names can only contain lowercase letters, numbers, hyphens, and underscores, and must be 21 characters or less.
  If you leave this field empty, function will try to use the slack_channel associated with the Incident or Task found in the Slack Conversations datatable. It there isn't one defined, the workflow will terminate.
- Slack is channel private indicates if the channel you are posting to should be private.
- Slack user emails is comma separated list of emails belonging to Slack users in your workspace that will be added to the channel you are posting to.
- Slack additional text message to include with the Incident, Note, Artifact, Attachment or Task data.

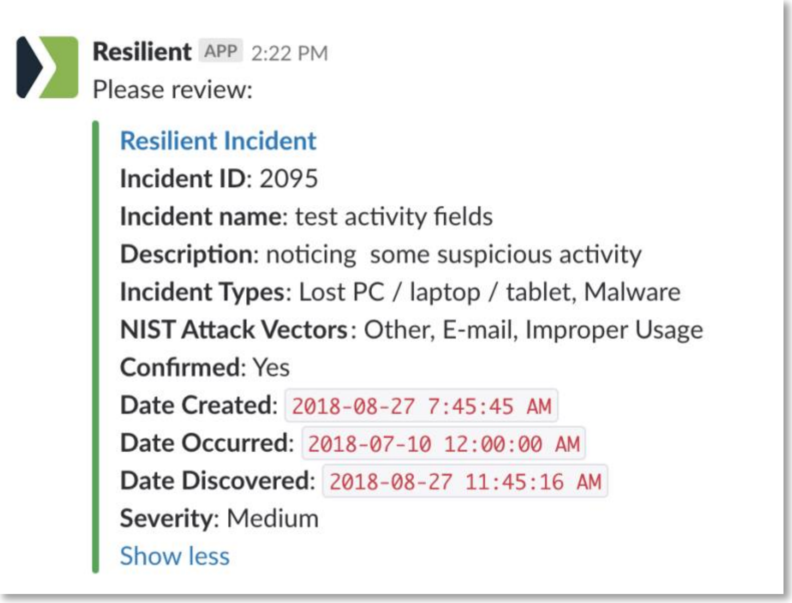
Before using a workflow

- Review the Incident, Task, Note and Artifact fields selected for posting to Slack in the workflow's Pre-Processing Script. A flexible JSON structure is used to define the Incident, Task, Note and Artifact fields to post to Slack using user-defined labels and formatting identifiers. Fields can be removed and added following the data structure defined.



```
# Slack text message in JSON format
# --------------------------------
# Do not remove first 3 elements "Additional Text", "Resilient URL" and "Type of data",
# the information is used to to generate the title of the message.
#
# Add/remove information using the syntax:
# "label": {{ "type": "[string|richtext|boolean|datetime", "data": "resilient field data" }}
# Make sure to send "datetime" types as int and not str -
# withouth duble quotes: { "type": "datetime", "data": resilient datetime data}
# watch out of embedded double quotes in text fields and escape with field.replace(u'"', u'\\"')
slack_text = u"""{{
  "Additional Text": {{"type": "string", "data": "{0}" }},
  "Resilient URL": {{"type": "incident", "data": "{1}" }},
  "Type of data": {{"type": "string", "data": "{2}" }},
  "Incident ID": {{"type": "string", "data": "{3}" }},
  "Incident name": {{"type": "string", "data": "{4}" }},
  "Description": {{"type": "richtext", "data": "{5}" }},
  "Incident Types": {{"type": "string", "data": "{6}" }},
  "NIST Attack Vectors": {{"type": "string", "data": "{7}" }},
  "Confirmed": {{"type": "boolean", "data": "{8}" }},
  "Date Created": {{"type": "datetime", "data": {9} }},
  "Date Occurred": {{"type": "datetime", "data": {10} }},
  "Date Discovered": {{"type": "datetime", "data": {11} }},
  "Severity": {{"type": "string", "data": "{12}" }}
}}""".format(
  rule_additional_text,
  incident_id,
  "Incident",
  incident_id,
  incident.name.replace(u'"', u'\\"'),
  description,
```

Your posted message will look like this in Slack:



Resilient APP 2:22 PM

Please review:

Resilient Incident

Incident ID: 2095

Incident name: test activity fields

Description: noticing some suspicious activity

Incident Types: Lost PC / laptop / tablet, Malware

NIST Attack Vectors: Other, E-mail, Improper Usage

Confirmed: Yes

Date Created: `2018-08-27 7:45:45 AM`

Date Occurred: `2018-07-10 12:00:00 AM`

Date Discovered: `2018-08-27 11:45:16 AM`

Severity: Medium

Show less

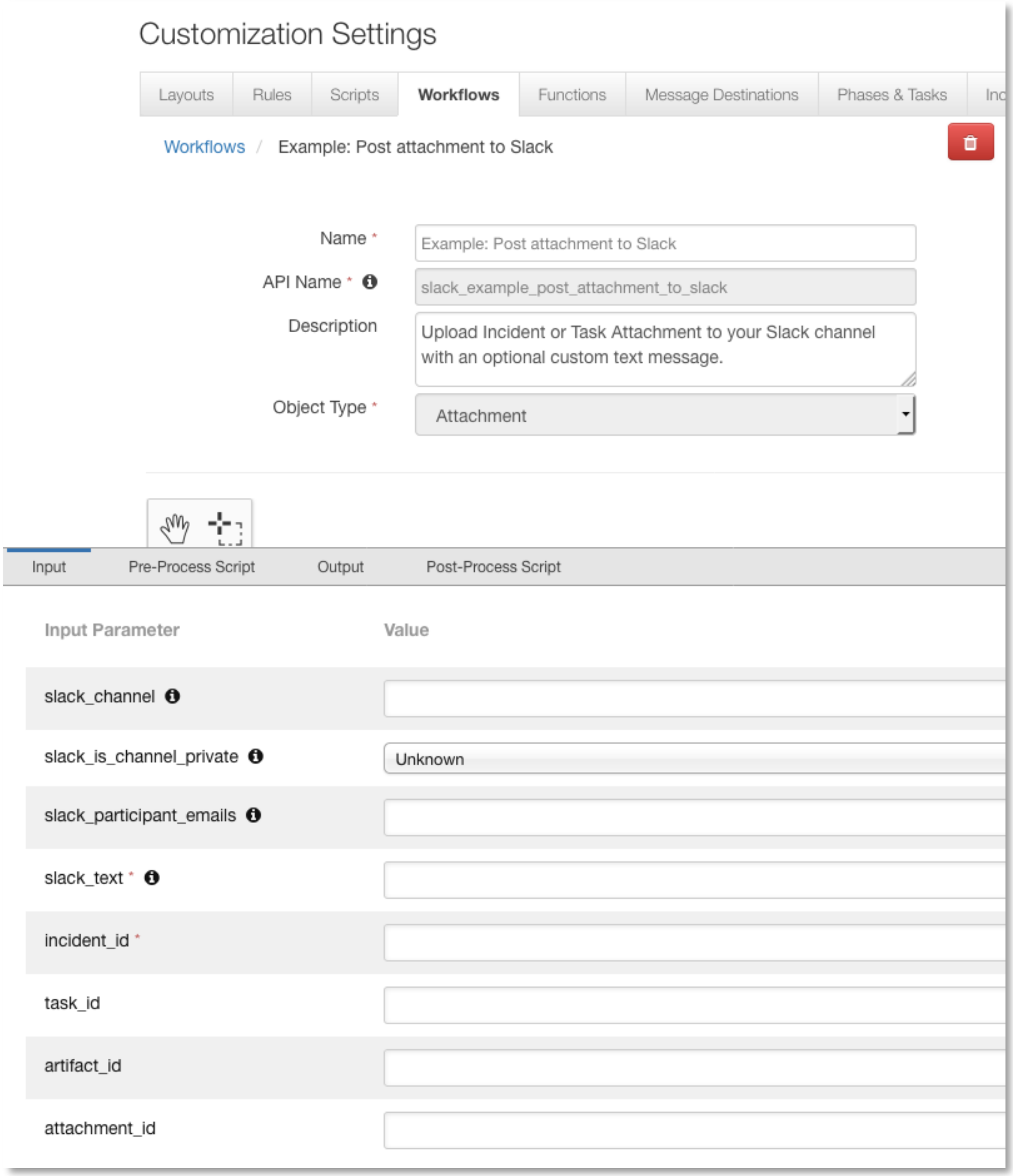
Fn_slack: slack_post_attachment

Function that uploads Incident, Task or Artifact attachments to Slack channel.

This function has two example workflows that demonstrate uploading Incident, Task, or Artifact Attachments to a chosen Slack channel.

You can see the input fields Slack function takes in Example: Post attachment to Slack and Example: Post attachment to Slack – Artifact.

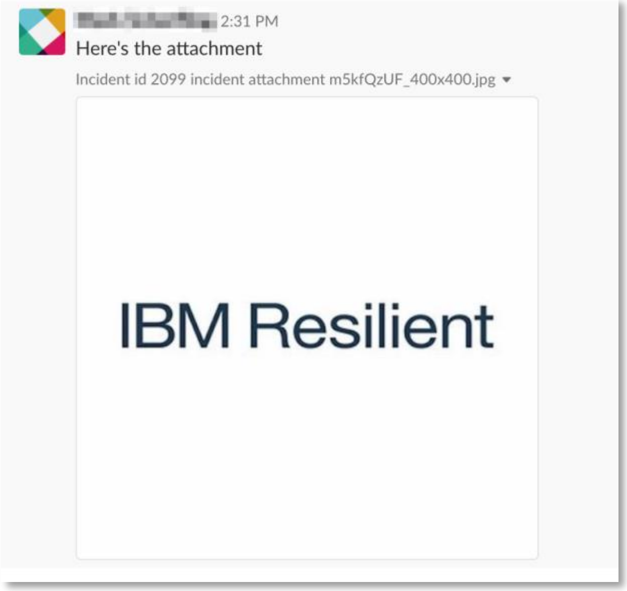
The function input fields start with `slack_` can be set on the Input Tab of the Workflow or when you click on the Menu Item, the same way as it's shown for the slack_post_message.



Name *	Example: Post attachment to Slack
API Name *	slack_example_post_attachment_to_slack
Description	Upload Incident or Task Attachment to your Slack channel with an optional custom text message.
Object Type *	Attachment

Input Parameter	Value
slack_channel	
slack_is_channel_private	Unknown
slack_participant_emails	
slack_text *	
incident_id *	
task_id	
artifact_id	
attachment_id	

Your posted message with a file upload will look like this in Slack:



[redacted] 2:31 PM
Here's the attachment
Incident id 2099 incident attachment m5kfQzUF_400x400.jpg ▾

To upload files we use [Slack API files.upload method](). Method uses Slack App OAuth Token that belongs to the authenticated user of the Slack App. File uploads sent using this token will be uploaded on behalf of the user - not the Slack App. This behavior can be changed by adding a [Bot User]() to your Slack App and using Bot User OAuth Token for file uploads.

Fn_slack: slack_archive_channel

Function that exports conversation history from Slack channel to a text file, saves the text file as a Resilient attachment and archives the Slack channel.

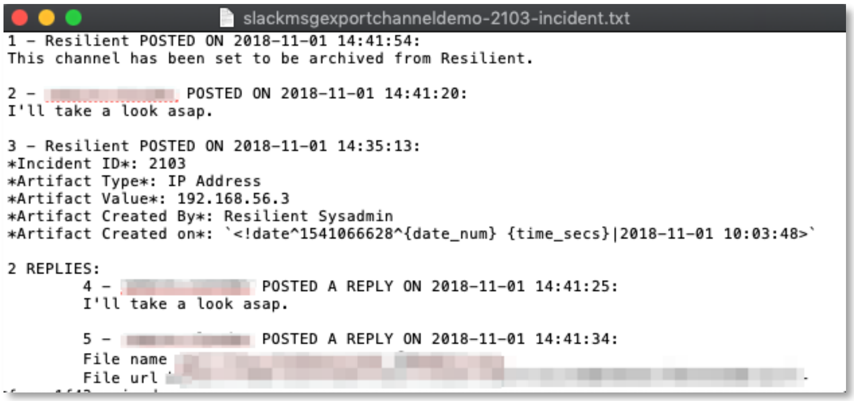
This function has two example workflows that demonstrate archiving Incident and Task.

You can see the input fields Slack function takes in Example: Archive Slack Channel – Incident and Example: Archive Slack Channel - Task.

The function input fields start with `slack_` can only be set on the Input Tab of the Workflow.

When invoking the Archive Slack Channel function, it will search for the associated channel_name in Slack Conversations Datatable. If Incident or Task have an existing connection with a slack_channel, function will archive this channel.

The exported conversation history looks like this:

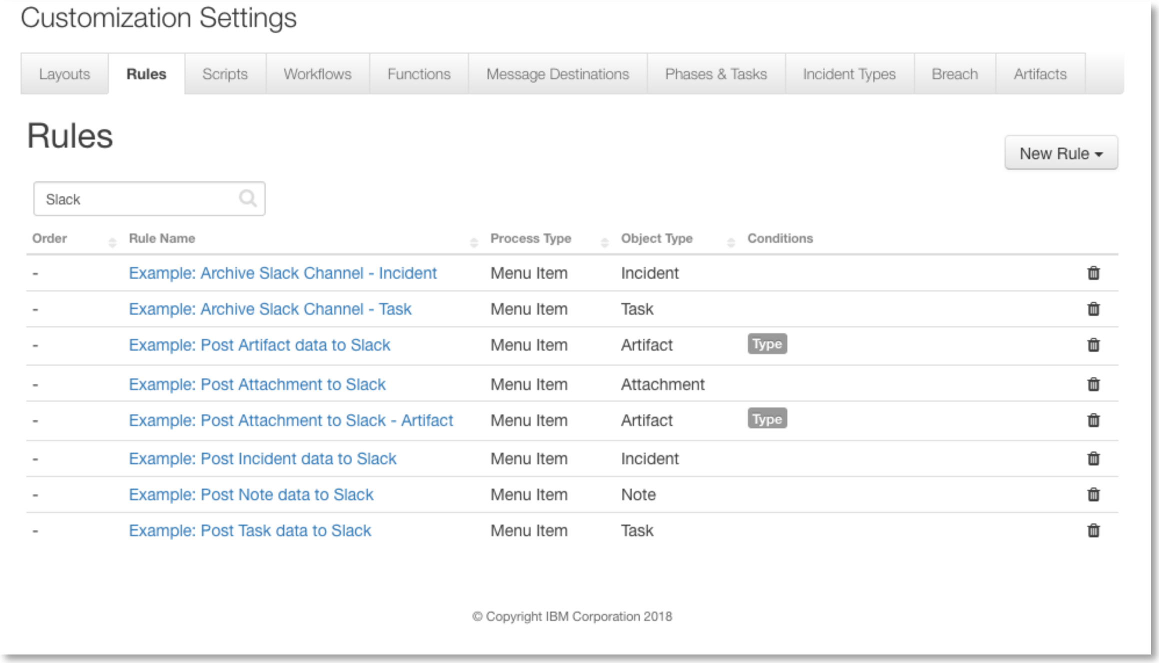


slackmsgexportchanneldemo-2103-incident.txt

```
1 - Resilient POSTED ON 2018-11-01 14:41:54:
This channel has been set to be archived from Resilient.

2 - [redacted] POSTED ON 2018-11-01 14:41:20:
I'll take a look asap.

3 - Resilient POSTED ON 2018-11-01 14:35:13:
*Incident ID*: 2103
*Artifact Type*: IP Address
*Artifact Value*: 192.168.56.3
*Artifact Created By*: Resilient Sysadmin
*Artifact Created on*: `<!date^1541066628^{date_num} {time_secs}|2018-11-01 10:03:48>`

2 REPLIES:
        4 - [redacted] POSTED A REPLY ON 2018-11-01 14:41:25:
        I'll take a look asap.

        5 - [redacted] POSTED A REPLY ON 2018-11-01 14:41:34:
        File name [redacted]
        File url [redacted]
```

Resilient Platform Configuration

Eight rules are defined which you can customize:

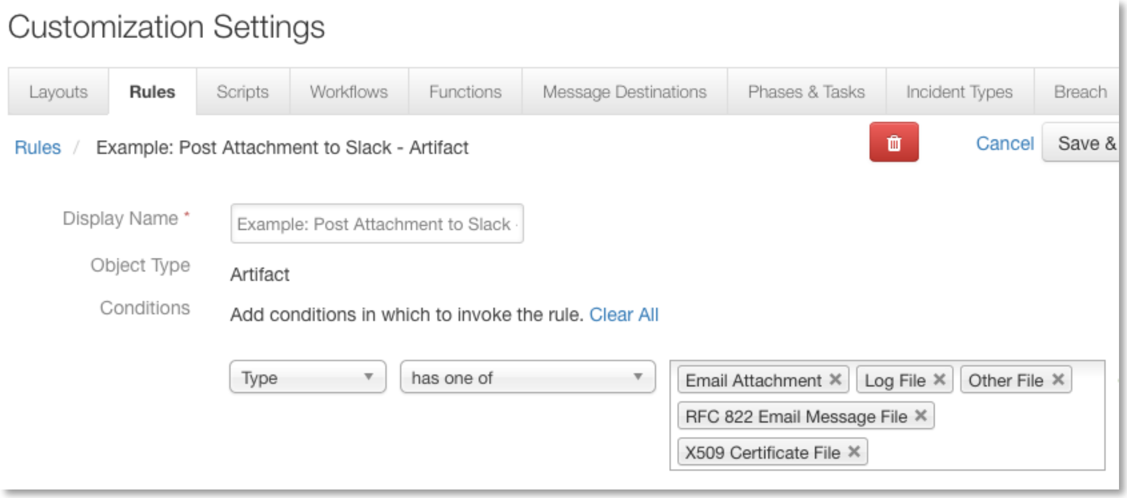


Rules

Order	Rule Name	Process Type	Object Type	Conditions
-	Example: Archive Slack Channel - Incident	Menu Item	Incident	
-	Example: Archive Slack Channel - Task	Menu Item	Task	
-	Example: Post Artifact data to Slack	Menu Item	Artifact	Type
-	Example: Post Attachment to Slack	Menu Item	Attachment	
-	Example: Post Attachment to Slack - Artifact	Menu Item	Artifact	Type
-	Example: Post Incident data to Slack	Menu Item	Incident	
-	Example: Post Note data to Slack	Menu Item	Note	
-	Example: Post Task data to Slack	Menu Item	Task	

© Copyright IBM Corporation 2018

Example: Post Attachment to Slack – Artifact rule allows uploading attachments from only certain types of artifacts. It's important to note that Artifact type `Malware sample` is excluded because of the potential risk of sending malware samples to your Slack channel.



Rules / Example: Post Attachment to Slack - Artifact

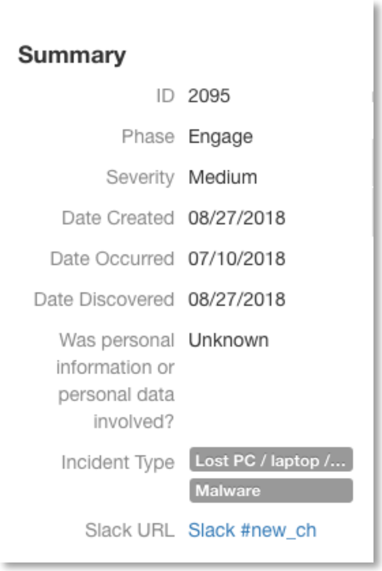
Display Name *: Example: Post Attachment to Slack ·

Object Type: Artifact

Conditions: Add conditions in which to invoke the rule. Clear All

Type ▾ | has one of ▾ | Email Attachment ✕ Log File ✕ Other File ✕ RFC 822 Email Message File ✕ X509 Certificate File ✕

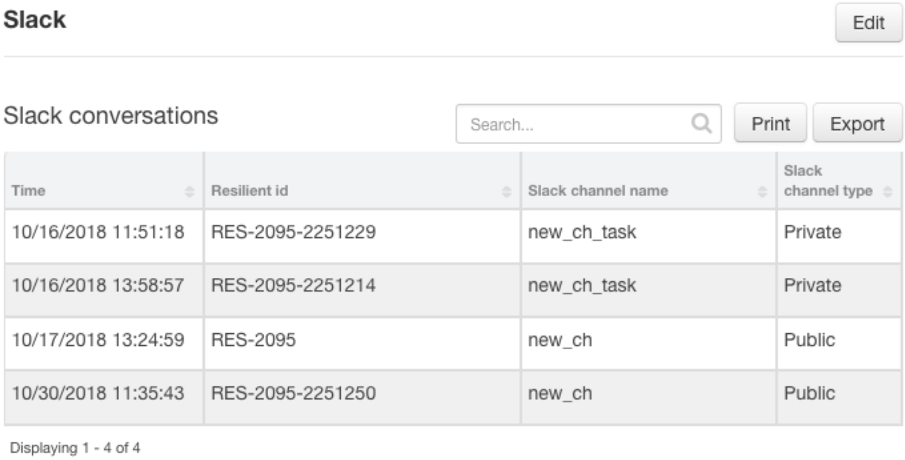
When you import fn_slack in the customize step it will create a custom incident field `slack_url` which is used in certain Workflow post-process scripts to save the URL to your Slack channel. Users may add this field to the Summary layout of the Incident.



Summary

ID	2095
Phase	Engage
Severity	Medium
Date Created	08/27/2018
Date Occurred	07/10/2018
Date Discovered	08/27/2018
Was personal information or personal data involved?	Unknown
Incident Type	Lost PC / laptop /... Malware
Slack URL	Slack #new_ch

There is also a Slack conversations datatable `slack_conversations_db` created in the customize step. Users may add this datatable to a custom layout.



Slack conversations

Time	Resilient id	Slack channel name	Slack channel type
10/16/2018 11:51:18	RES-2095-2251229	new_ch_task	Private
10/16/2018 13:58:57	RES-2095-2251214	new_ch_task	Private
10/17/2018 13:24:59	RES-2095	new_ch	Public
10/30/2018 11:35:43	RES-2095-2251250	new_ch	Public

Displaying 1 - 4 of 4

The purpose of the datatable is to save connections between an Incident and a Slack channel or a Task and a Slack channel. Incident and Tasks can each have one associated channel – the default channel, but users may also post to other channels – separate special channels, if they wish to. The function only saves the connection to the first Slack channel user posted in.

When invoking the Post message to Slack or Post attachment to Slack functions, users can specify whether they are posting to their default Slack channel or to a separate special channel.

If the channel_name function input field, either on the Menu Item or on the Input Tab on Workflow, is not specified, function will search for the associated channel_name in Slack Conversations Datatable. If Incident or Task have an existing connection with a slack_channel, function will find this channel in your Workspace and use it to post in.

If channel_name input field is specified, function will try finding this channel in your Workspace to use it to post in or if it doesn't exist it will create a new one to post in.

If channel_name input field is specified and there is also an associated channel – the default channel found in datatable, function will ignore the associated one and post to the input one – separate special channel.

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.