IBM Resilient

resilient

Incident Response Platform Integrations

Slack Function V1.0.0

Release Date: May 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 Function.

Overview

Slack is an online communications solution allowing communities to communicate as groups or directly with each other through text channels and video conferences. This Resilient platform functions-based integration allows incidents and notes to be shared with these Slack text channels. It can be used to extend the communication about incidents to additional groups and individuals.

Two workflows are present which allow incident data and notes to be shared. Both of these workflows utilize a single function which posts new messages and adds replies to existing conversation threads.

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

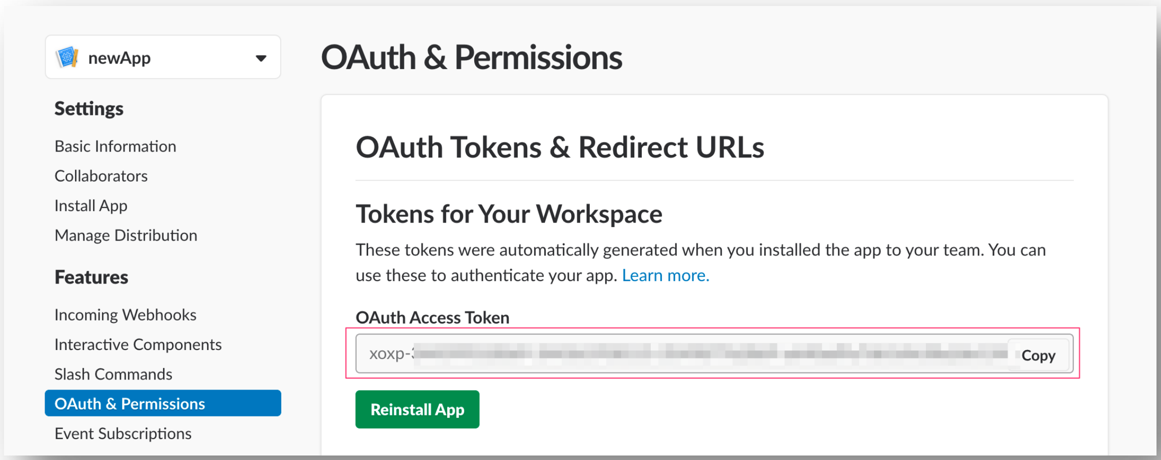
Installation

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.)

Slack configuration

Prior to configuring the Resilient functional processor, follow the Slack documentation (https://api.slack.com/slack-apps) on building a new app for external bot. There is a section titled OAuth and Permissions which contains 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-xxxxxxx-xxxxxxxx-xxxxxxxxxxx

   # Name for source of messages as displayed in Slack
   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.

Deploy these customizations to the Resilient platform with the following command:

```
resilient-circuits customize
```

Answer 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 recommended 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.

The unit file should be named 'resilient_circuits.service':

```
sudo vi /etc/systemd/system/resilient_circuits.service
```

The contents:

```
[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
```

Ensure that the service unit file is correctly permissioned:

```
sudo chmod 664 /etc/systemd/system/resilient_circuits.service
```

Use the systemctl command to manually start, stop, restart and return status on the service:

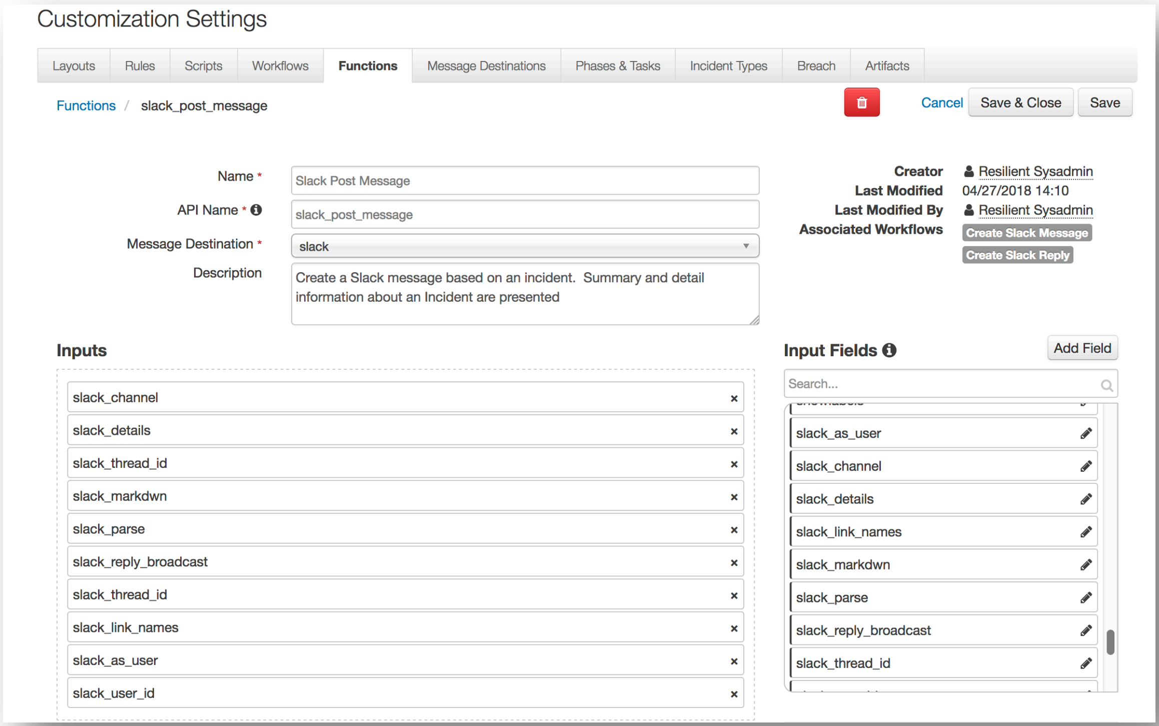
```
sudo systemctl resilient_circuits [start|stop|restart|status]
```

Log files for systemd and the resilient-circuits service can be viewed through the journalctl command:

```
sudo journalctl -u resilient_circuits --since "2 hours ago"
```

Function Descriptions

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



Name * Slack Post Message

API Name * slack_post_message

Message Destination * slack

Description: Create a Slack message based on an incident. Summary and detail information about an Incident are presented

Creator: Resilient Sysadmin

Last Modified: 04/27/2018 14:10

Last Modified By: Resilient Sysadmin

Associated Workflows: Create Slack Message, Create Slack Reply

Inputs:

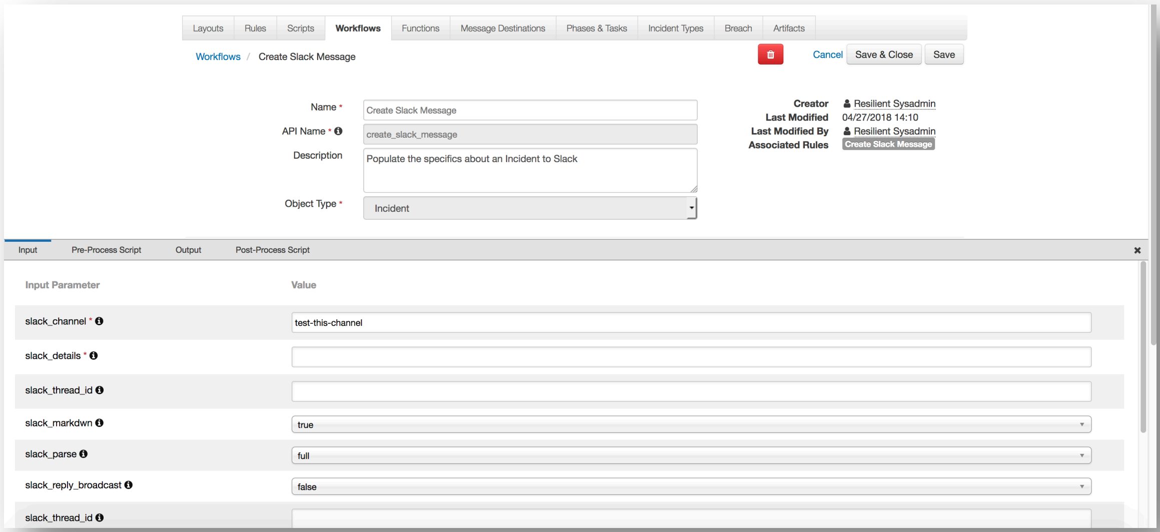
- slack_channel
- slack_details
- slack_thread_id
- slack_markdwn
- slack_parse
- slack_reply_broadcast
- slack_thread_id
- slack_link_names
- slack_as_user
- slack_user_id

Input Fields:

- slack_as_user
- slack_channel
- slack_details
- slack_link_names
- slack_markdwn
- slack_parse
- slack_reply_broadcast
- slack_thread_id

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. Refer to the Slack API documentation on the use of the slack arguments such as slack_markdwn, slack_parse, etc.

See the Slack function in the workflows: Create Slack Message and Create Slack Reply. Review the Input tab when editing the function within a workflow for the default settings.



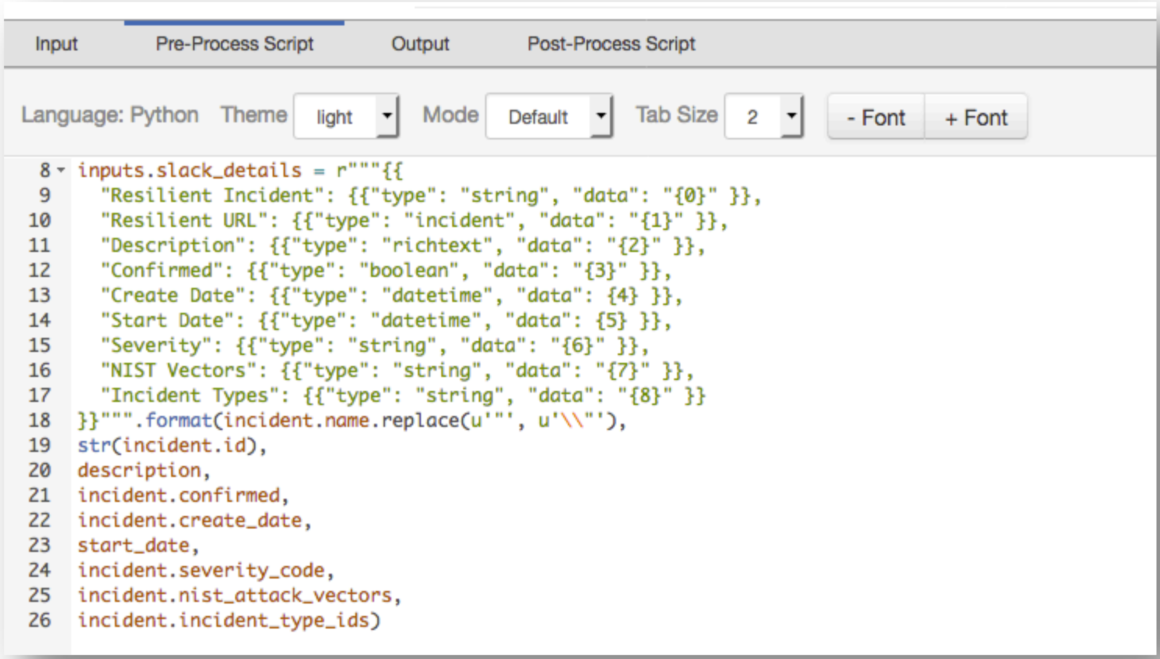
Name * Create Slack Message
API Name * create_slack_message
Description Populate the specifics about an Incident to Slack
Object Type * Incident

Creator Resilient Sysadmin
Last Modified 04/27/2018 14:10
Last Modified By Resilient Sysadmin
Associated Rules Create Slack Message

Input Parameter	Value
slack_channel *	test-this-channel
slack_details *	
slack_thread_id	
slack_markdwn	true
slack_parse	full
slack_reply_broadcast	false
slack_thread_id	

Before using a workflow

- Change the defined slack_channel in the Input tab for your environment.
- Review the Incident and Note fields selected for posting to Slack in the workflow's Pre-Processing Script. A flexible JSON structure is used to define the incident, task and note fields to post to Slack using user-defined labels and formatting identifiers. Fields can be removed and added following the data structure defined.

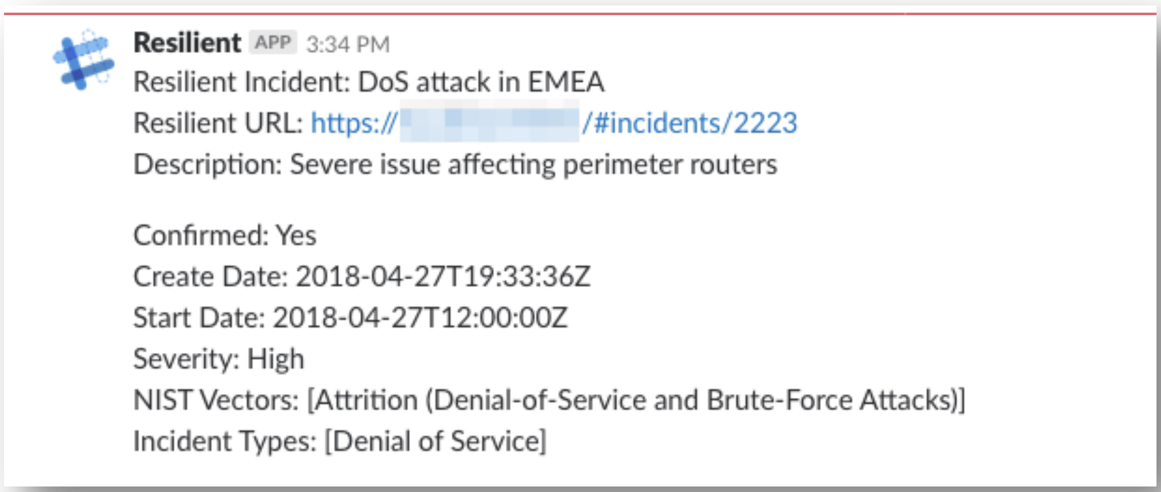


```python
inputs.slack_details = r"""{{
  "Resilient Incident": {{"type": "string", "data": "{0}" }},
  "Resilient URL": {{"type": "incident", "data": "{1}" }},
  "Description": {{"type": "richtext", "data": "{2}" }},
  "Confirmed": {{"type": "boolean", "data": "{3}" }},
  "Create Date": {{"type": "datetime", "data": {4} }},
  "Start Date": {{"type": "datetime", "data": {5} }},
  "Severity": {{"type": "string", "data": "{6}" }},
  "NIST Vectors": {{"type": "string", "data": "{7}" }},
  "Incident Types": {{"type": "string", "data": "{8}" }}
}}""".format(incident.name.replace(u'"', u'\\"'),
str(incident.id),
description,
incident.confirmed,
incident.create_date,
start_date,
incident.severity_code,
incident.nist_attack_vectors,
incident.incident_type_ids)
```

Resilient Platform Configuration

Two rules are defined which you can customize: Create Slack Message and Create Slack Reply.

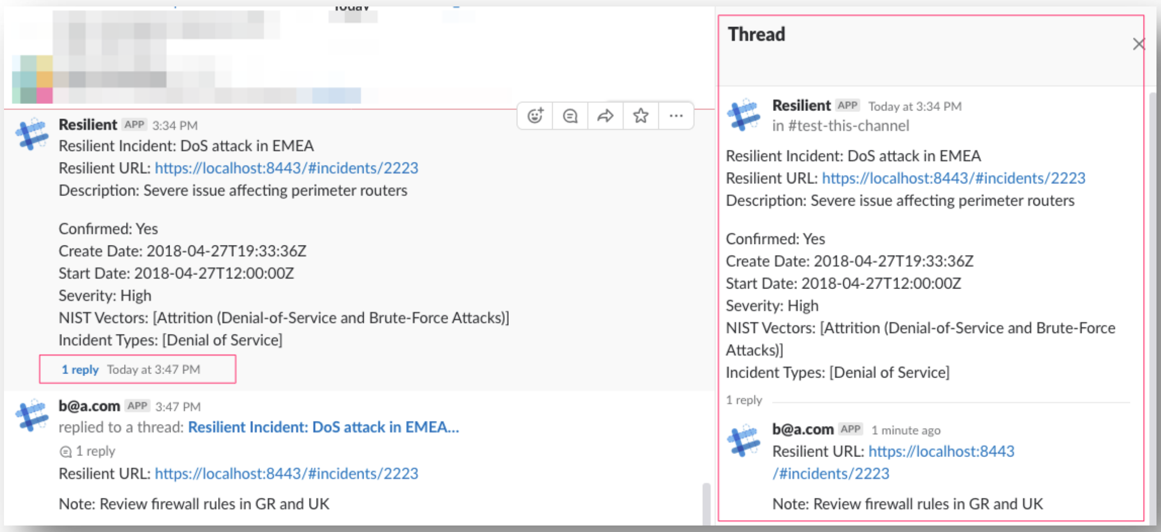
Create Slack Message – this is a manual rule that is run from an incident's Action menu. A Slack thread ID is retained to create threaded conversations when notes are added (see the use of the input field slack_thread_id).



Resilient APP 3:34 PM
Resilient Incident: DoS attack in EMEA
Resilient URL: https://[redacted]/#incidents/2223
Description: Severe issue affecting perimeter routers

Confirmed: Yes
Create Date: 2018-04-27T19:33:36Z
Start Date: 2018-04-27T12:00:00Z
Severity: High
NIST Vectors: [Attrition (Denial-of-Service and Brute-Force Attacks)]
Incident Types: [Denial of Service]

Create Slack Reply – this is an automatic rule that is triggered when a note is added to an incident already posted to Slack. When the slack_thread_id is used, the slack display shows the new note and its reply to the posted incident conversation.



[redacted]

Resilient APP 3:34 PM
Resilient Incident: DoS attack in EMEA
Resilient URL: https://localhost:8443/#incidents/2223
Description: Severe issue affecting perimeter routers

Confirmed: Yes
Create Date: 2018-04-27T19:33:36Z
Start Date: 2018-04-27T12:00:00Z
Severity: High
NIST Vectors: [Attrition (Denial-of-Service and Brute-Force Attacks)]
Incident Types: [Denial of Service]
1 reply Today at 3:47 PM

b@a.com APP 3:47 PM
replied to a thread: Resilient Incident: DoS attack in EMEA...
1 reply
Resilient URL: https://localhost:8443/#incidents/2223

Note: Review firewall rules in GR and UK

Thread

Resilient APP Today at 3:34 PM
in #test-this-channel

Resilient Incident: DoS attack in EMEA
Resilient URL: https://localhost:8443/#incidents/2223
Description: Severe issue affecting perimeter routers

Confirmed: Yes
Create Date: 2018-04-27T19:33:36Z
Start Date: 2018-04-27T12:00:00Z
Severity: High
NIST Vectors: [Attrition (Denial-of-Service and Brute-Force Attacks)]
Incident Types: [Denial of Service]

1 reply

b@a.com APP 1 minute ago
Resilient URL: https://localhost:8443/#incidents/2223

Note: Review firewall rules in GR and UK

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.