IBM Resilient

Incident Response Platform Integrations

Jira Function V1.0.2

Release Date: September 2019

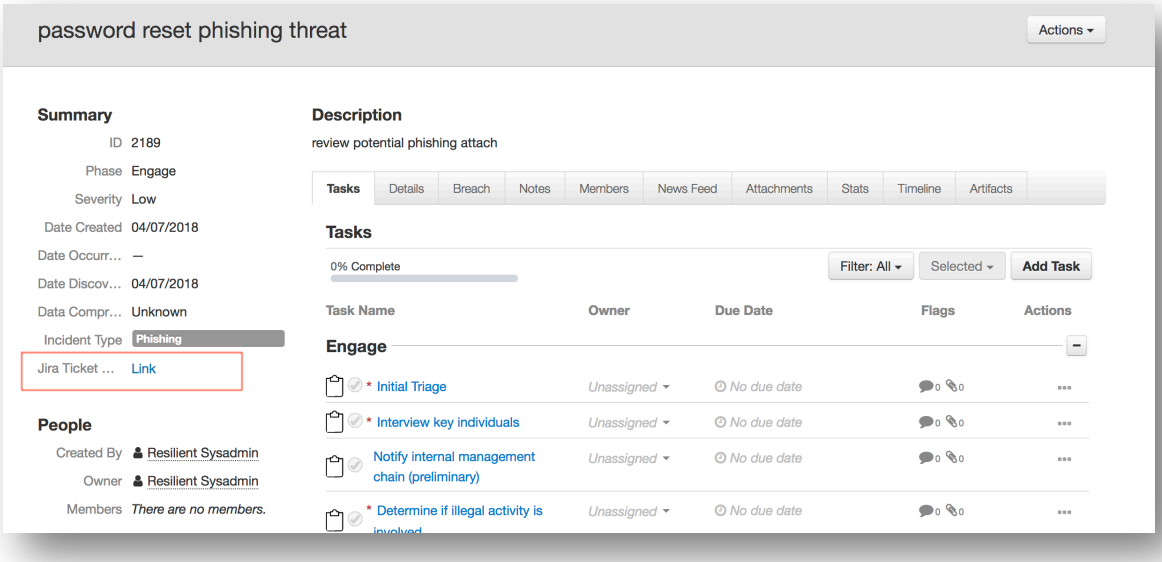
Resilient Functions simplify development of the integrations by sending data from the Resilient platform to a remote program that performs an activity then returns the results to the function. The results can be acted upon by a script and the result of that becomes a decision point in the Resilient workflow.

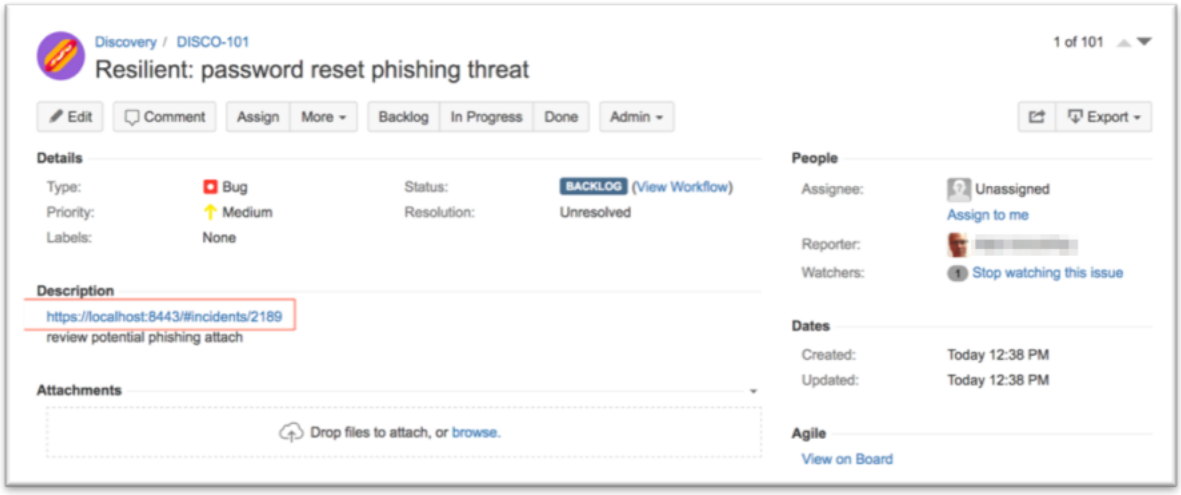
This guide describes the Jira Function Integrations.

Overview

The Jira integration with the Resilient platform allows for the tracking of Incidents as Jira Issues. Bidirectional links are saved to allow for easy navigation between the applications.

This integration allows for the creation of Jira issues, adding comments, and transitioning issues when the corresponding Incident is closed. The following screenshots show how this would appear in an incident.





Setup

The following lists the system requirements:

- Python version 2.7.10 or later, or version 3.6 or later
- Resilient Circuits and Resilient Python libraries version 30.0 or later
- Resilient platform version 31.0 or later
- Jira version 7.4 or later

Perform the following to install and configure the function:

1. Ensure the environment is up to date:
   ```
   sudo pip install --upgrade pip
   sudo pip install --upgrade setuptools
   sudo pip install --upgrade resilient-circuits
   ```
2. Install the required software for the function (if not already installed):
   ```
   sudo pip install fn_jira-<version>.tar.gz
   ```
3. Add the function to the Resilient platform:
   ```
   resilient-circuits customize
   ```
   You are prompted to answer prompts to import functions, message destinations, etc.
4. From the account used for Integrations, use one of the following commands to configure the Jira settings. Use –c for new environments or –u for existing environments.
   ```
   resilient-circuits config -c
   ```
   OR
   ```
   resilient-circuits config -u
   ```
5. Edit the .resilient/app.config file and section [jira]:
   ```
   url=<jira url>
   user=<jira access user>
   password=<jira access password>
   verify_cert=[True|False]
   ```
   Use False for self-signed SSL certificates.

After completing the configuration steps, enter the `resilient-circuits run` command. The following is an example of the resulting messages indicating the successful connection to the Resilient platform and the loading of the Jira integration modules.

```
$ resilient-circuits run

2018-04-07 12:38:04,164 INFO [app] Configuration file:
/Users/Integration/.resilient/app.config

2018-04-07 12:38:04,165 INFO [app] Resilient server: <host>

2018-04-07 12:38:04,165 INFO [app] Resilient user: <acct>

2018-04-07 12:38:04,165 INFO [app] Resilient org: <org>

2018-04-07 12:38:04,165 INFO [app] Logging Level: INFO

…

2018-04-07 12:38:05,418 INFO [component_loader]
'fn_jira.components.jira_open_issue.FunctionComponent' loading

2018-04-07 12:38:05,419 INFO [component_loader]
'fn_jira.components.jira_create_comment.FunctionComponent' loading

2018-04-07 12:38:05,420 INFO [component_loader]
'fn_jira.components.jira_transition_issue.FunctionComponent' loading

…

2018-04-07 12:38:05,435 INFO [actions_component]
'fn_jira.components.jira_open_issue.FunctionComponent' function 'jira_open_issue'
registered to 'jira'

2018-04-07 12:38:05,436 INFO [actions_component]
'fn_jira.components.jira_create_comment.FunctionComponent' function
'jira_create_comment' registered to 'jira'

2018-04-07 12:38:05,437 INFO [actions_component]
'fn_jira.components.jira_transition_issue.FunctionComponent' function
'jira_transition_issue' registered to 'jira'

…

2018-04-07 12:38:05,729 INFO [actions_component] Subscribe to message destination
'jira'

…

2018-04-07 12:38:05,731 INFO [stomp_component] Subscribe to message destination
actions.<org id>.jira

…
```

Resilient Platform Configuration

In the Customization Settings section of the Resilient platform, you can verify that the following Jira specific functions, workflows and rules are available in the Resilient platform by clicking their respective tabs.

- Functions
  - **Jira Create Issue**. Creates a Jira issue including an incident's title, description and severity level. A custom Incident field provides a link back to Jira. See the corresponding workflows which defines which issue project and issue type to use.
  - **Jira Create Comment**. Creates a Jira comment based on adding a note to an incident. The note's description field is referenced.
  - **Jira Transition Issue**. Transitions a Jira issue. See the corresponding workflow which defines the transition ID to use.

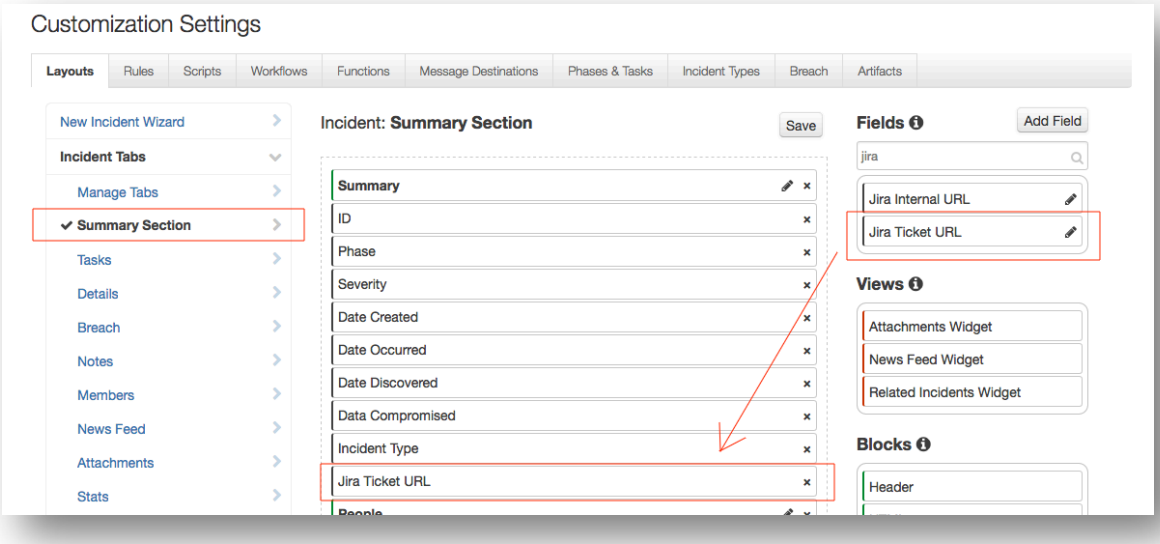
- Workflows. Some modifications are needed for your Jira environment as indicated below.
    - **Example: Jira Open Issue**. Edit the Jira Open Issue function for the specific `jira_project` and `jira_issuetype` fields for your Jira use.
    - **Example: Jira Open Issue (Task)**. Similar to creating a Jira issue from an incident, but instead creates a Jira issue based on an incident task.
    - **Example: Jira Create Comment**.
    - **Example: Jira Transition Issue**. Edit the Jira Transition Issue function for the specific `jira_transition_id` required in your workflow. The default value of 41 represent the Jira close issue state.
- Rules. Operate on an incident, task or an incident's notes. If you wish to change rules from automatic to menu item, new rules referencing the same workflows need to be created.
    - Example: Create Jira Issue
    - Example: Create Jira Issue (Task)
    - Example: Jira Create Comment
    - Example: Jira Close Issue
    - Example: Jira Close Issue (Task)

Layout

Jira Ticket URL Custom Field

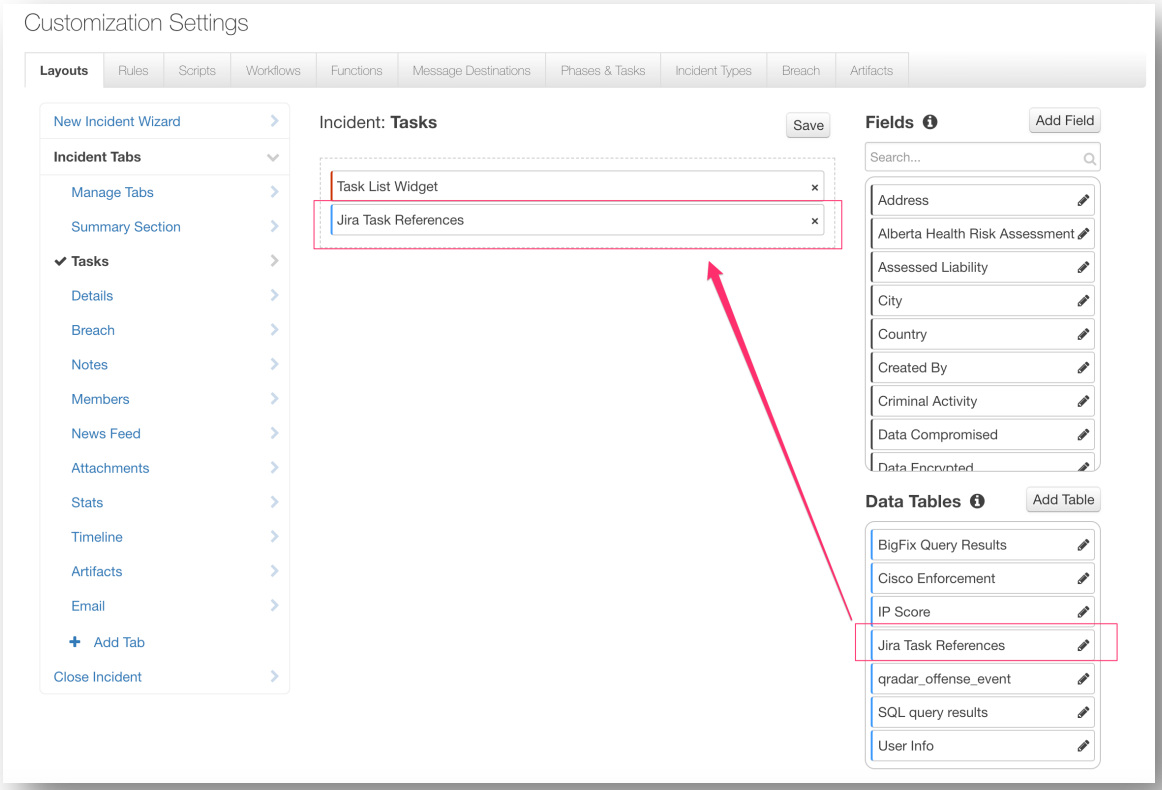
To display a link back to the Jira issue created from an incident,

- Navigate to the Customization Settings and select the Summary Section from the Layouts tab
- Search for the Jira custom fields and drag Jira Ticket URL to the Summary Section
- Click Save



Jira Task Reference Data Table

- Perform the same update operations with the Jira Task References data table on the Task layout.



Operation

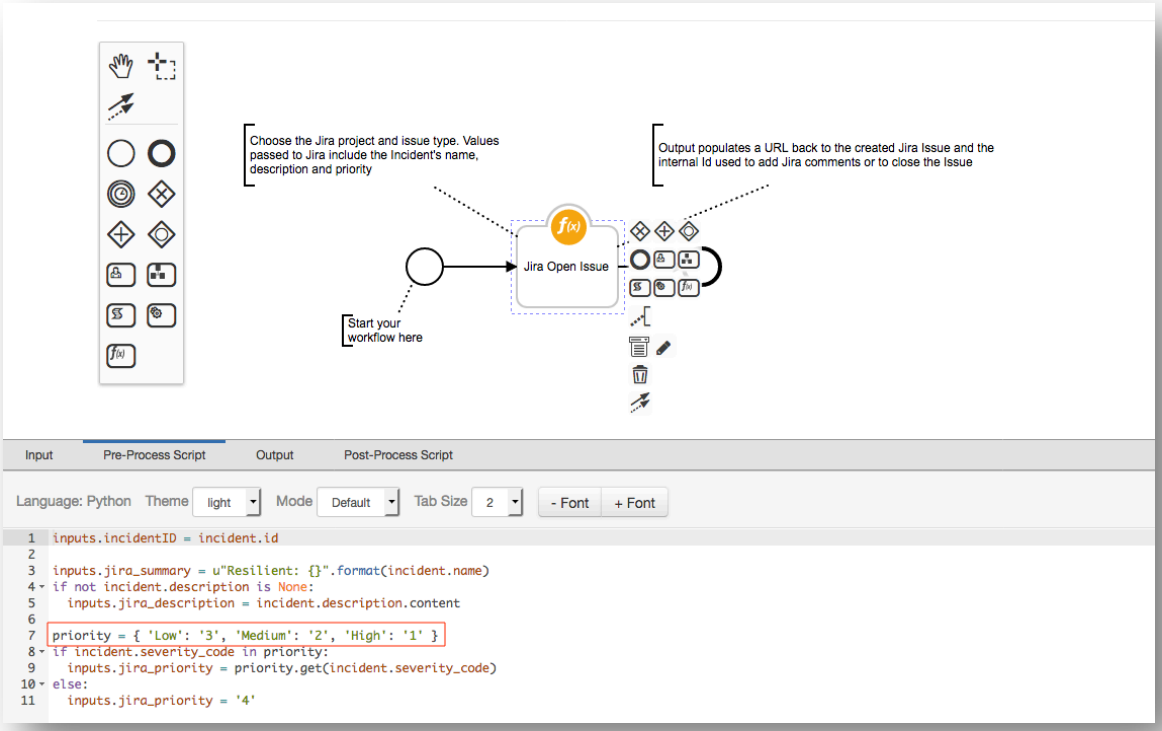
There are two use cases provided in this package as reflected in the example rules. Your use cases will vary, and it is suggested that you define your own workflows and rules for production use.

1) Create a Jira issue associated with a Resilient incident
    a. Automatically push Resilient comments to Jira after the Jira issue is created
    b. Automatically close the Jira issue when the Resilient incident is closed
2) Create a Jira issue associated with an incident's task
    a. Automatically push Resilient comments to Jira after the Jira issue is created
    b. Automatically close the Jira issue when the Resilient incident is closed

Modifications

Severity and Priority Mapping

Both the Jira Open Issue and Jira Transition Issue workflows use a mapping between Resilient incident severity codes and Jira issue priorities. If you wish to have a different mapping to Jira, modify these two workflows and their corresponding Pre-Processing Scripts.



Assignee

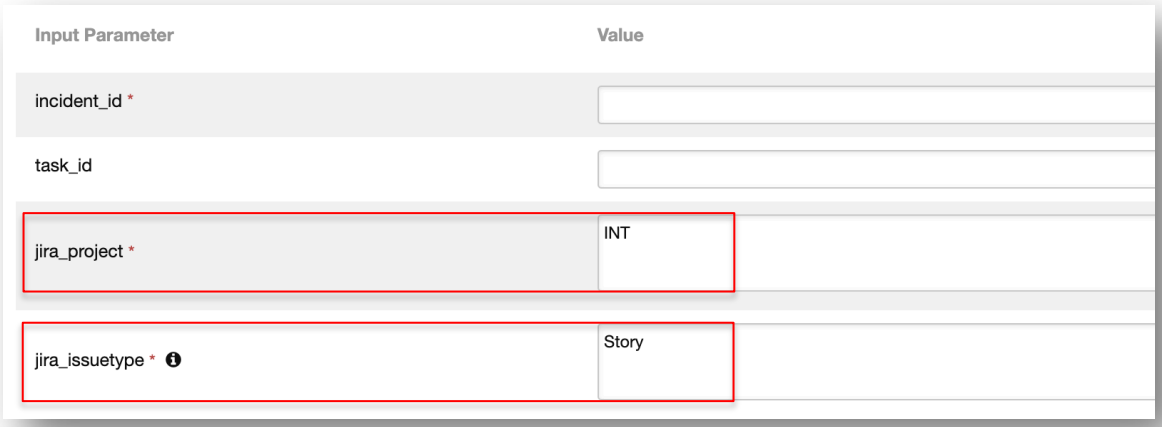
The assignee of the Jira ticket is controlled by a pre-processing script lookup relationship between Resilient users and Jira users. Edit the following line in 'Example: Jira Open Issue' workflow to match the list associated with your enterprise:

```
lookup = { "Resilient email ID": "JIRA assignee" [, "user2": "jira user2" ] }
```

Refer to the post-processing scripts for comments on the sample payload returned.

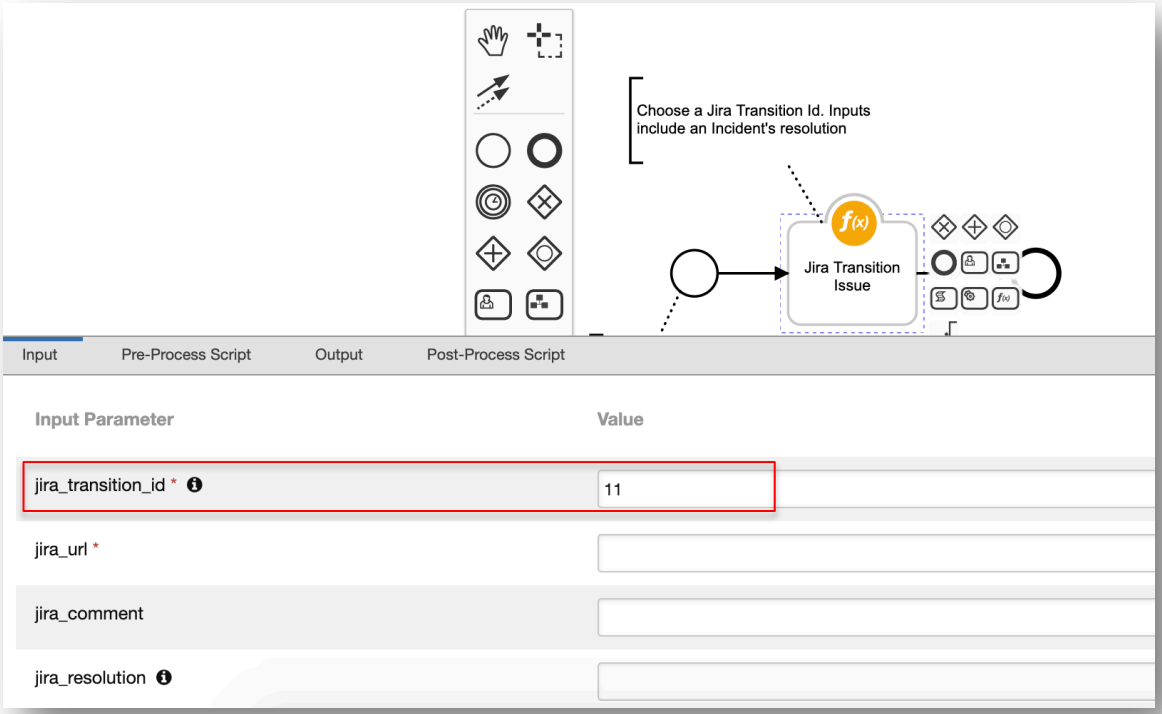
Jira Project Type and Issue Type

For the workflows which create Jira issues, the input parameters to the Jira Open Issue function define which Jira Project and Issue Type to create. The default values should be changed to reflect the specific project and issues for your environment.

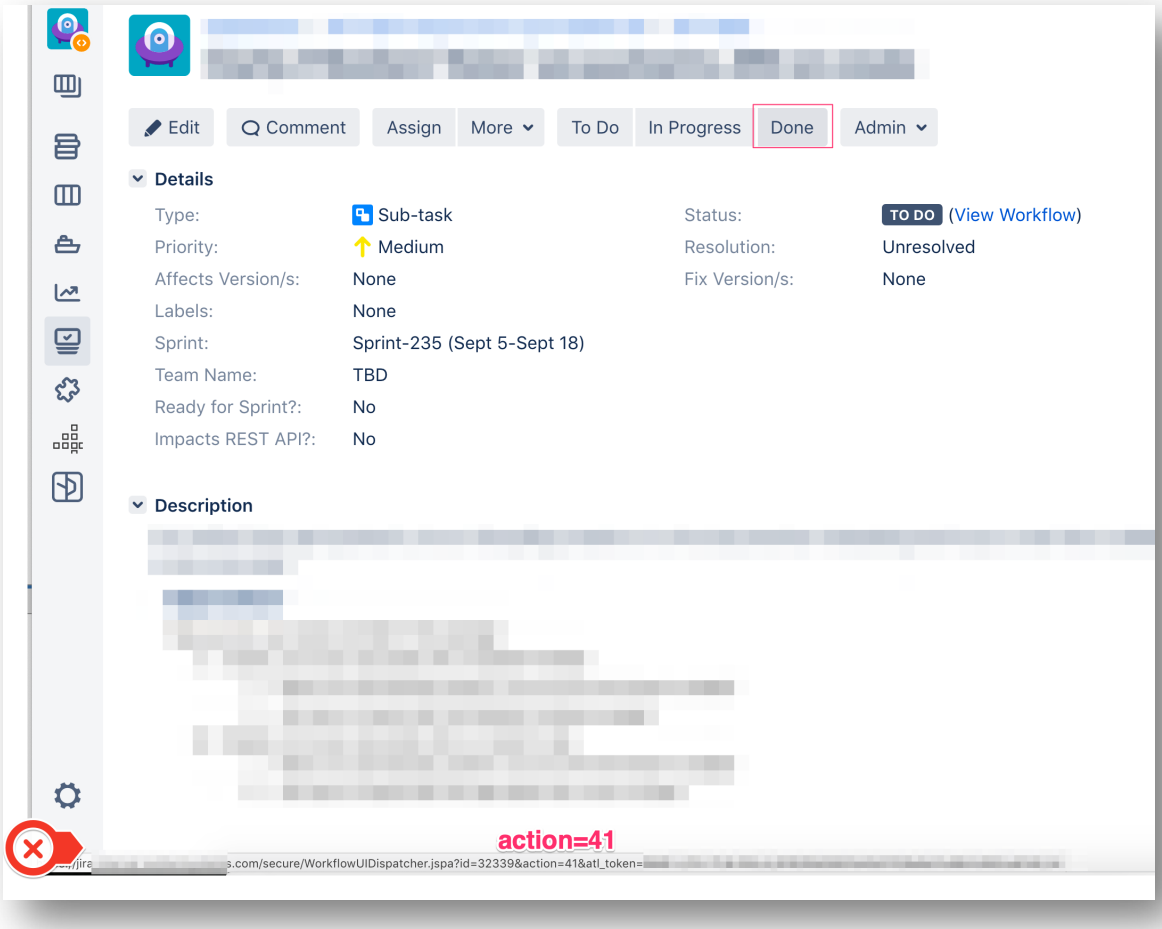


Jira Issue Transitions

When a Resilient incident is closed, it's possible to also close (or change state) on the Jira issue. This is controlled by the Input Parameter for the Jira Function: jira_transition_id. This value should match the value valid for your Jira issue workflow.

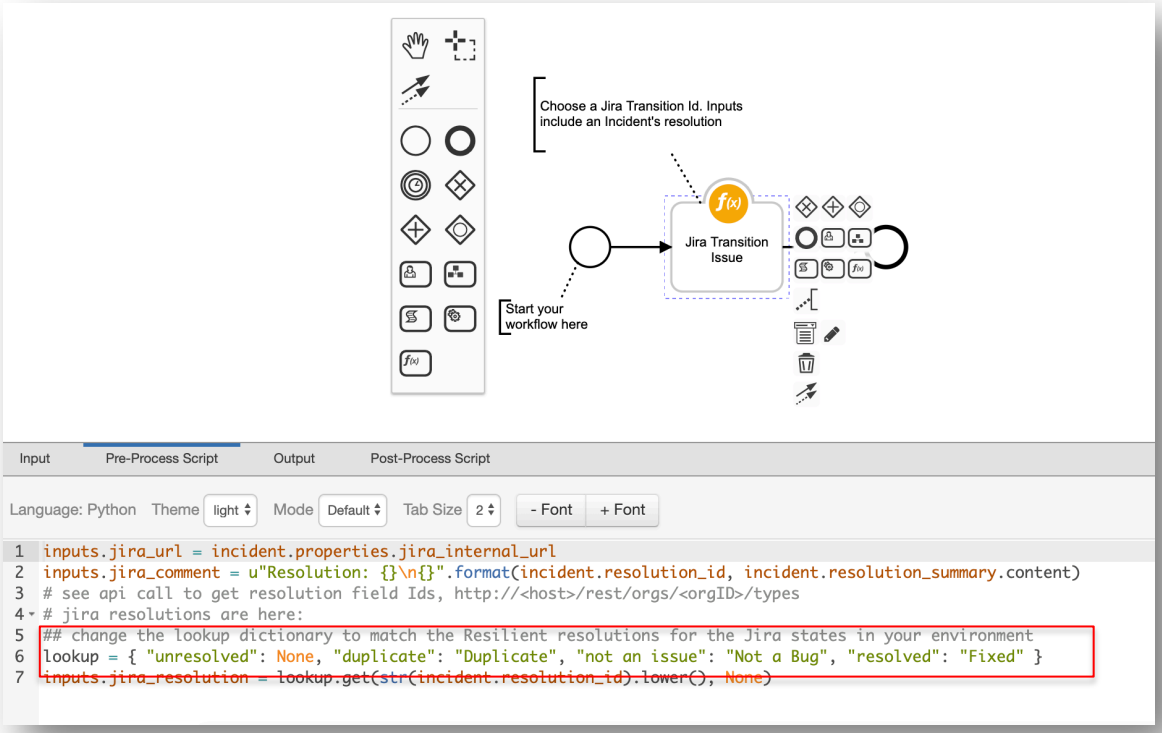


The value to use can be found in Jira through their API for a given project type. One simple technique is to hover over the Jira State button, such as Done, and see the action value in the display of the URL to submit (see below).



Jira Resolution

Some Jira issue types can support a resolution field when the issue is closed. This field can be mapped to the incident's Resolution field. See the lookup table defined in the pre-processor script of the Example: Jira Transition Issue and modify the values appropriately to match the possible incident resolution values and those values used in your Jira issue.



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 creates progress information. Failures show up as errors and may contain Python trace statements.

Support

For additional support, go to Resilient Community Forum at https://ibm.biz/resilientcommunity.