Meshlet Configuration for Jira Service Desk

Configuration File

A configuration file is needed for each meshlet to customize the integration (application.yml).

The default incident configuration file is as follows:

Default Incident Configuration File

```
spring:
    rabbitmq:
        host: localhost
        port: 5672
        username: admin
        password: adminadmin
        vhost: /
perspectium:
    meshlet:
        scheduler:
            pollingInterval: 30000
        topic: siam
        type: common_incident
        key: jira_sd_meshlet
    jira:
        url: https://example.atlassian.net/rest/api/2/issue/
        projectKey: IT
        server: true
        query: updated >= '-5m' and type = 'incident'
        workNotesPrivate: true
        user:
            watcher:
                addWatchers: true
                watchersField: additional_assignee_list
            missingUser:
                createMissingUsers: true
            userArrayField:
                addUserArrayField: true
                userArraySourceField: examplefield
                userArrayTargetField: examplefield
        transition:
            useTransitions: true
            transitionMap:
                Closed: 91
                '[In Progress]': 31 # '[]' notation used to maintain whitespace in key
    message:
        inboundQueue: psp.in.meshlet.jiraservicedesk.incident.key
        outboundQueue: psp.out.servicenow.devexample
        errorQueuePattern: psp.out.meshlet.jiraservicedesk.incident.error
    auth:
        username: exampleuser
        password: examplepass
    directory: ${application_directory}
```

While most or all of these values will be configured by Perspectium Support, the following is a list of the important configurable keys:

Key	Description
url	The url of your JSD instance REST API.
projectKey	The key of the JSD project to integrate to.
watcher. addWatchers	True if using a watcher field. A watcher field is a field that will be mapped to the watchers field in JSD.
watcher. watchersField	The name of the watcher field in the Common Document. The value of the field must be a comma separated list of usernames (e.g. *robert.fan,john.smith,hsimpson*).

missingUser.createMissingUsers	True if you want to create a JSD user that exists in a field but does not exist in JSD. Requires a display name, username, and email address, which are provided via fields in the Common Document. See the default mapping *createUser.json.*
userArrayField.addUserArrayField	True if you have a "user array" field that you want to map. This type of field is a special field in JSD that takes a list of users. The list of users must be a comma separated list of usernames (e.g. *robert.fan,john.smith,hsimpson*) and must all exist in JSD.
userArrayField.addUserArraySourceField	The source field for the "user array" field in the Common Document.
userArrayField.addUserArrayTargetField	The target field for the "user array" field in JSD.
transition.useTransitions	True if you are using JSD transitions in your integration. To do this you are required to provide a transition map.
transition.transitionMap	A map of transition ids to use for each status change. The key is the name of the status (i.e. the value of the field mapped to JSD status in the Common Document). In JSD, each status change requires a transition which is identified by a transition id. In the example configuration file above, if the status changes to *Closed* then we use the transition with id 91.
auth.username	The JSD integration user username.
auth.password	The JSD integration user password.

Mappings

The JSD meshlet comes with a set of default mappings which can be configured to your liking. Value mappings are also available, which allow a user to map certain values in the inbound Common Document to another arbitrary value.

An example value mapping file:

Example Value Mapping

```
{
         "1": "Critical",
         "2": "High",
        "3": "Medium",
        "4": "Low"
}
```

This file maps the priority field from ServiceNow (formatted as an integer by default) to a priority field in JSD (formatted as a String), which solves the issue of converting between the two different formats.

The following is an example of a valueMapping.json file configured for the above mapping and another reversed value mapping:

Example valueMapping.json

```
[
        {
           "Source":"common_incident",
           "Target":"issue",
           "Field":"priority",
           "Description":"Mapping for common incident to case severity",
           "Mapping":"${file:static/values/ci_to_issue_priority.json}"
        },
        {
           "Source":"issue",
           "Target":"common_incident",
           "Field":"priority",
           "Description":"Mapping for issue priority to common_incident priority",
           "Mapping":"${file:static/values/issue_to_ci_priority.json}"
        }
]
```

Every time a value mapping is added (by adding a value mapping file) a corresponding entry must be created in the valueMapping.json file. When creating another entry you can follow the format of the example valueMapping.json file above.

Trying to map your AWS Support Center cases to fields in another app?

Contact Perspectium Support for a guided setup.