

ServiceNow Integration with JIRA

A) When an Issue is created/updated in JIRA a record will automatically created in a custom table of ServiceNow.

1.Create a custom application/table in ServiceNow.

The screenshot shows the 'Jira Staging Table' form in ServiceNow. The form has a header bar with a back arrow, a menu icon, the title 'Jira Staging Table', and a 'New record' link. On the right side of the header bar are icons for help, settings, and a 'Submit' button. The form fields are arranged in two columns: Project Key, Issue Key, Issue Id, Summary, Project Name, and Description. A 'Submit' button is located at the bottom left of the form.

2.Create a Scripted REST API.

All>System Webservices>Scripted Webservices>Scripted REST APIs

The screenshot shows the 'Scripted REST Service' configuration page in ServiceNow. The header bar includes a back arrow, a menu icon, the title 'Scripted REST Service', and a subtitle 'Jira to servicenow'. On the right are icons for help, settings, and buttons for 'Update', 'Delete', and a dropdown arrow. The form fields include: Name (Jira to servicenow), API ID (jiratoservicenow), Active (checkbox), Protection policy (None), Application (Global), API namespace (962943), and Base API path (/api/962943/jiratoservicenow). Below the form is a 'Security' tab with a 'Content Negotiation' sub-tab. A blue box contains text about Default ACLs. A red arrow points from a text box to the 'Default ACLs' dropdown menu, which is set to 'Scripted REST External Default'.

Default ACL will restrict the users having **snc_external** role to excute the REST Endpoint.

3.Create a POST Resource in Scripted REST API.

The screenshot shows the 'Scripted REST Resource' configuration page in ServiceNow. The header bar includes a back arrow, a menu icon, the title 'Scripted REST Resource', and a subtitle 'POST'. On the right are icons for help, settings, and buttons for 'Update', 'Delete', and a dropdown arrow. The form fields include: API definition (Jira to servicenow), Name (POST), Application (Global), Active (checkbox), HTTP method (POST | POST), Relative path (/), and Resource path (/api/962943/jiratoservicenow). A blue box contains text about Request routing. A red arrow points from a text box to the 'HTTP method' dropdown menu, which is set to 'POST'.

Select HTTP Method as POST.

HTTP Methods:

- 1.GET: used to retrieve data from any third party application.
- 2.POST: used to create records in any third party application.

- 3.PUT:** used to update specified fields but other fields will be updated with the default values.
- 4.PATCH:** used to update specified fields and other fields will not get updated with the default values.
- 5.DELETE:** used to delete any data from any third party application.

4. Write the script.

```
Script
1  (function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {
2
3      try {
4          //gs.log("Jiralog :"+ JSON.stringify(request.body.data));
5          var requestBody = request.body.data;
6          //gs.log('jiralog '+ requestBody.issue.fields.project.name);
7
8          var jr= new GlideRecord('u_jira_stage');
9          jr.addQuery('u_issue_id',requestBody.issue.id);
10         jr.addQuery('u_project_key',requestBody.issue.fields.project.key);
11         jr.query();
12         if(!jr.next())
13         {
14
15             jr.initialize();
16             jr.u_project_name = requestBody.issue.fields.project.name;
17             jr.u_project_key = requestBody.issue.fields.project.key;
18             jr.u_issue_id = requestBody.issue.id;
19             jr.u_issue_key = requestBody.issue.key;
20             jr.u_summary= requestBody.issue.fields.summary;
21             jr.u_description=requestBody.issue.fields.description;
22             jr.insert();
23             gs.log('Jira Issue created: '+jr.u_issue_id);
24         }
25
26         else{
27
28             jr.u_summary= requestBody.issue.fields.summary;
29             jr.u_description=requestBody.issue.fields.description;
30             jr.update();
31             gs.log('Jira Issue updated: '+jr.u_issue_id);
32
33         }
34     }
35
36     catch(ex) {
37         var message = ex.message;
38     }
39
40
41 })(request, response);
```

5. Create an API Token for Authorization.

Go to this URL and create an API Token then copy it.

<https://id.atlassian.com/manage-profile/security/api-tokens>

The screenshot shows the 'API Tokens' management interface. At the top, there are two buttons: 'Create API token' (highlighted in blue) and 'Revoke all API tokens' (in grey). Below these, a text box explains that API tokens should be treated as secure and that a maximum of 25 tokens can be active at once. A second text box notes that new tokens may take up to a minute to become active. The interface is divided into two main sections. The right section, titled 'Create an API token', prompts the user to 'Choose a label that is short, memorable, and easy for you to remember.' It features a text input field for the label, a 'Cancel' button, and a 'Create' button. A red box and callout '2. Give a Label.' highlight this section. The left section, titled 'Your new API token', displays a masked token (a series of dots) with an eye icon to toggle visibility. Below the token is a 'Copy' button and a 'Close' button. A red box and callout '3. Copy API Token.' highlight this section. A third callout, '1. Click on create api token.', points to the 'Create API token' button at the top.

6. Create a user in ServiceNow

***NOTE:** Username will be your JIRA login email id & password will be API token we get in step 5 and give that user **web_service_admin** role.

7. Register a Webhook in JIRA

Webhook: it is a user defined callback over HTTP. We can use JIRA Webhook to notify application whenever certain events occur in JIRA.

Example: You might have to alert your remote application when an issue is updated or when any sprint is started use a Webhook to do this means that your remote application does not have to periodically poll JIRA via REST APIs to determine changes have occurred.

Steps:

- Go to JIRA Administrators Console > System (under settings) > Webhooks (In advance section).

- Create a Webhook.
- Enter details.

Name: Servicenow Integration

URL: Enter your scripted REST API URL.

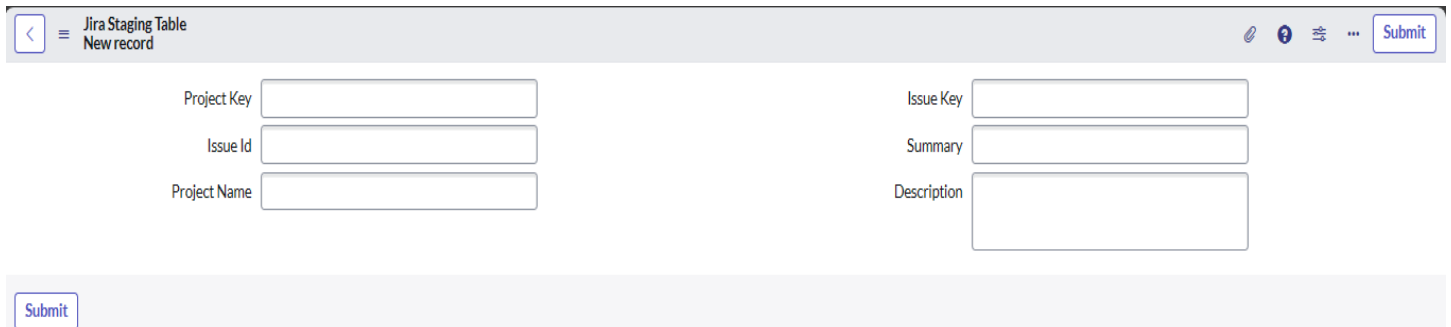
Events: Issue Created

- Create.

7. When you create an issue in JIRA it will create a record in your custom table.

B) When a record is created in custom table it will automatically create an issue in JIRA.

1.Create a custom application/table in ServiceNow.



The screenshot shows the 'Jira Staging Table' form in ServiceNow. The form has a header bar with a back arrow, a menu icon, the title 'Jira Staging Table', a subtitle 'New record', and a 'Submit' button. The form body contains six input fields arranged in two columns. The left column has 'Project Key', 'Issue Id', and 'Project Name'. The right column has 'Issue Key', 'Summary', and 'Description'. A 'Submit' button is located at the bottom left of the form.

2.Create a REST Message

Check these links to get an Idea for what purpose what HTTP method has to be used ,URL & Payload format.

- [Jira REST API examples \(atlassian.com\)](https://developer.atlassian.com/cloud/jira/platform/rest-api-examples/)
- [The Jira Cloud platform REST API \(atlassian.com\)](https://developer.atlassian.com/cloud/jira/platform/rest-api/)

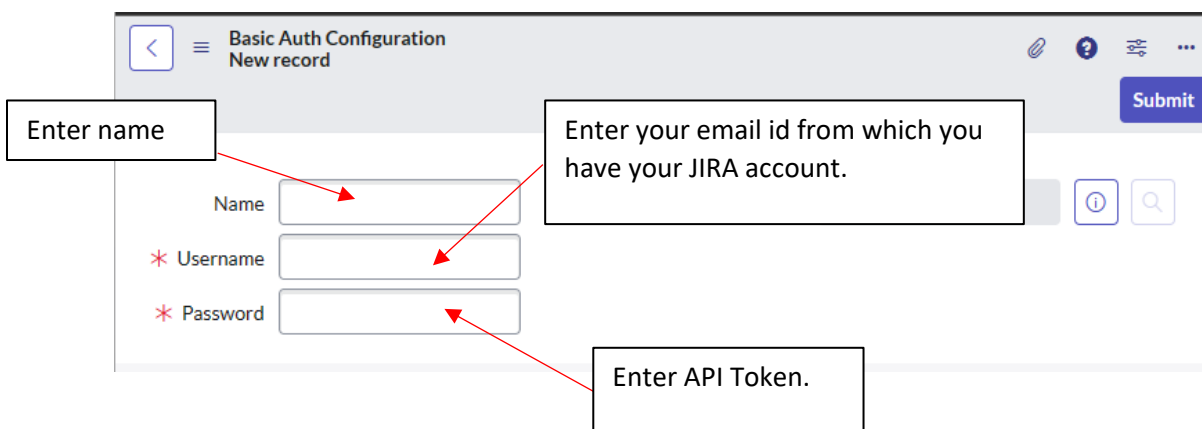
All>System Webservices > Outbond > REST Message

- EndPoint: <https://your-domain.atlassian.net/rest/api/3/issue>

replace your-domain in URL with your atlassian url.

- Authentication type:Basic

Basic Auth Profile:



The screenshot shows the 'Basic Auth Configuration' form in ServiceNow. The form has a header bar with a back arrow, a menu icon, the title 'Basic Auth Configuration', a subtitle 'New record', and a 'Submit' button. The form body contains four input fields: 'Name', 'Username', 'Password', and 'API Token'. There are three callout boxes with red arrows pointing to the fields: 'Enter name' points to the 'Name' field, 'Enter your email id from which you have your JIRA account.' points to the 'Username' field, and 'Enter API Token.' points to the 'API Token' field. The 'Password' field is marked with a red asterisk.

***NOTE:**Process of getting an API Token is above in Step 5.

3.Create a POST HTTP Method.

HTTP Method POST

Select method as POST.

REST Message Jira Integration

* Name POST

* HTTP method POST | post

Endpoint https://void06.atlassian.net/rest/api/3/issue

Update Delete

HTTP Header:

Authentication HTTP Request

Use MID Server

HTTP Headers

Name	Value
Accept	application/json
Content-Type	application/json
Insert a new row...	

To get the script of this REST Message

By Clicking on this related Link you can get check your connection is OK or NOT.

Related Links

- [Auto-generate variables](#)
- [Preview Script Usage](#)
- [Set HTTP Log level](#)
- [Test](#)

Click on this Related Link and copy the script.

HTTP response status codes;

- 100-199
- 200-299
- 300-399
- 400-499
- 500-599

Informational responses
Successful responses
Redirection messages
Client error responses
Server error responses

4.This is the example of PayLoad Data which is accepted in JIRA(JSON Format).

```
{
  "fields": {
    "project": {
      "key": "Your Project Name"
    },
    "issuetype": {
      "name": "Task"
    },
    "summary": "This issue is created via REST Message by Abhijeet",
    "description": {
      "type": "doc",
      "version": 1,
      "content": [{
        "type": "paragraph",
        "content": [{
          "type": "text",
          "text": "This is testing for description by Abhijeet Singh"
        }]
      }]
    }
  }
}
```

4.Create a Business Rule so that whenever a record is created in the custom table script will run.

All > System Definition > Business Rules

Business Rule
Servicenow to Jira

A business rule is a server-side script that runs when a record is displayed, inserted, deleted, or when a table is queried. Use business rules to automatically change values in form fields when the specified conditions are met. [More Info](#)

Name:

Table: Select Custom Table .

Application:

Active: ☒

Advanced: ☒

When to run: Select Insert After Because this BR will only after the record is Inserted in the Database.

Order:

Filter Conditions:

Role conditions:

Insert: ☒
Update: ☐
Delete: ☐
Query: ☐

And write the script in the advance section.

```

2
3
4   try {
5       var r = new sn_ws.RESTMessageV2('Jira Integration', 'POST');
6       var body = {
7           "fields": {
8               "project": {
9                   "key": current.u_project_key.toString()
10              },
11              "issuetype": {
12                  "name": "Task"
13              },
14              "summary": current.u_summary.toString()
15              "description": {
16                  "type": "doc",
17                  "version": 1,
18                  "content": [{
19                      "type": "paragraph",
20                      "content": [{
21                          "type": "text",
22                          "text": current.u_description.toString()
23                      }]
24                  }]
25              }
26          }
27      };
28      gs.log("testingBody"+body);
29
30      r.setRequestHeader(JSON.stringify(body));
31      var response = r.execute();
32      var responseBody = response.getBody();
33      var httpStatus = response.getStatusCode();
34      gs.addInfoMessage("Testing"+responseBody);
35      gs.addInfoMessage("Testing"+httpStatus);
36  } catch (ex) {
37      var message = ex.message;
38  }
39
40  })(current, previous);

```

add .toString() because in JSON data can only be send in string format.

5. When you create a record it will automatically create an Issue in JIRA.