# Sample ServiceNow as a Ticketing System

**Create a REST Connection for ServiceNow**:

* Set up a REST connection in Saviynt EIC to communicate with ServiceNow.
* Use OAuth2 or Basic Authentication for secure API requests, as configured in the connection JSON.

**Select ServiceNow as the ServiceDesk Connection**:

* Configure ServiceNow as the ServiceDesk connection for the third-party application where you need to track and resolve tasks.
* This setup allows you to trigger ticket creation in ServiceNow when tasks (like creating accounts, adding access, or removing access) are generated in EIC.

**Generate Tickets on Task Creation in EIC**:

* When a task is created in EIC, a corresponding ticket is automatically generated in ServiceNow.
* This ticket is created through the CreateTicketJSON configured in EIC, using the WSRETRY provisioning job to synchronize the task and ticket.

**Ticket Management in ServiceNow**:

* ServiceNow administrators manage tickets, checking for open tickets, updating their statuses, and performing closure actions upon completion.
* Once a task is fulfilled in the target third-party application, the admin can close the ticket in ServiceNow.

**EIC Polls for Ticket Status**:

* EIC periodically checks the ticket status in ServiceNow using the TicketStatusJSON configuration.
* The status of tickets can be 'Closed,' 'Open,' or 'Pending,' which helps EIC determine the task's progress.

**Status Synchronization**:

* When ServiceNow marks a ticket as 'Closed,' EIC detects the status change on its next polling cycle.
* EIC then updates the status of the corresponding task to 'Completed,' ensuring consistency between the systems.

## Configuration Steps

Below is the Connection JSON configuration needed to establish a REST connection with the ServiceNow application:

{  
 "authentications": {  
 "userAuth": {  
 "authType": "oauth2",  
 "url": "URL",  
 "httpMethod": "POST",  
 "httpHeaders": {  
 "contentType": "application/x-www-form-urlencoded"  
 },  
 "httpContentType": "application/x-www-form-urlencoded",  
 "expiryError": "ExpiredAuthenticationToken",  
 "authError": [  
 "InvalidAuthenticationToken",  
 "AuthenticationFailed"  
 ],  
 "timeOutError": "Read timed out",  
 "errorPath": "error.code",  
 "maxRefreshTryCount": 5,  
 "tokenResponsePath": "access\_token",  
 "tokenType": "Basic",  
 "accessToken": "Basic token"  
 }  
 }  
}

## Explanation of Configuration Parameters

* authentications:

Indicates the authentication JSON parameters, which can be of types such as basic, token generation using Contrast, and token re-generation.

* acctAuth:

Name of the connection.

* authType:

Represents the authentication type. Supported types include Basic, Application Contrast, and Token Refresh based on your requirements.

* URL:

Indicates the URL to be used for Contrast or Token Refresh authentication types.

* httpMethod:

Indicates the HTTP method to be used (GET/POST) for performing authentication.

* httpHeaders:

HTTP headers required for connection with the target system.

* httpContentType:

Indicates the content type of the HTTP request, typically 'application/x-www-form-urlencoded' for OAuth2.

* authError:

Defines the error response from the target application for different types of authentication failures.

* errorPath:

Error path defines the path of error in the response JSON file.

* maxRefreshTryCount:

The maximum number of times to refresh the token.

* tokenResponsePath:

Path of the access token in the response JSON.

* tokenType:

Token Type indicates whether it is Basic or Bearer.

* accessToken:

Valid access token to be specified, which can be a Base64 token, OAuth token, or accessToken for token refresh.

## Examples of Authentication Types

Examples for different authentication types are provided below:  
  
1. \*\*Basic Authentication\*\*  
 - "properties": {  
 "userName": "IAM.USER",  
 "password": "Welcome@1"  
 }  
  
2. \*\*Token Refresh\*\*  
 - URL: https://<url>/login  
  
3. \*\*Application Contrast Authentication\*\*  
 - URL: https://app.contrastsecurity.com/Contrast/api/ng/<Org-UID>/users

Create Ticket Json:  
Specify this parameter to create tickets in ServiceNow.

When you run the WSRETRY job using the CREATETICKETJSON parameter, it creates tickets in ServiceNow. The possible status of tickets in ServiceNow are: Open, Closed, or Pending. The ticket remains in Open status in ServiceNow till it is automatically or manually completed in ServiceNow.

Example 1: To define this parameter, use a format similar to the following:

JSON

{

"call": [

{

"name": "call1",

"connection": "userAuth",

"url": "https://<domainname>/api/now/table/sc\_task",

"httpMethod": "POST",

"httpParams": "{\"bp\_id\":\"$user.username\",\"name\":\"$user.lastname, $user.firstname\",\"email\":\"$user.email\",\"permissions\": \"${allEntitlementsValues}\"}",

"httpHeaders": {

"Authorization": "${access\_token}" },

"httpContentType": "application/json",

"ticketidPath": "Request Number",

"unsuccessResponses": {

"message": "" }

}

]

}

To configure the TicketStatusJSON parameter for checking and updating ticket statuses in ServiceNow, follow the JSON structure below. This setup uses the WSRETRY job to poll ServiceNow for ticket statuses, and based on the status retrieved, updates the task status in EIC accordingly.

Here's a sample JSON configuration:

json

Copy code

{

"call": [

{

"name": "call1",

"connection": "userAuth",

"url": "https://<domain-name>/api/now/table/sc\_req\_item?sysparm\_query=request.number=${ticketID}&sysparm\_limit=1&sysparm\_display\_value=true",

"httpMethod": "GET",

"httpHeaders": {

"Authorization": "${access\_token}"

},

"httpContentType": "application/json",

"ticketStatusPath": "result[0].state",

"ticketStatusValue": [

"Closed",

"CLOSED",

"closed"

],

"successResponses": [

{}

]

}

]

}

**Parameter Definitions**

* **ticketStatusPath**: Specifies the JSON path in the API response where the ticket status is located. In this example, "result[0].state" is where ServiceNow's API response provides the ticket status.
* **ticketStatusValue**: Lists the possible values for a ticket's "Closed" status, ensuring all potential case variations are captured (e.g., Closed, CLOSED, closed). If any of these values are returned by ServiceNow, EIC will update the task status to "Completed."

The following steps provide the entire workflow of how a request is created and closed in EIC after the ticket is closed in ServiceNow.

1. Log in to EIC.
2. Go to ARS and submit the type of request (create account, remove account, or add access, remove access).
3. Approve the request in EIC.
4. Run the WSRETRY job. The ticket is created in ServiceNow using the CreateTicket JSON.
5. The admin logs in to the ServiceNow Ticketing system and checks the open tickets.
6. Based on the ServiceNow managed application to which the access is requested, the admin completes the request in that application and closes the ticket in ServiceNow.
7. The connector polls ServiceNow using the TicketStatus JSON to check the status of the ticket.
8. When the ticket is closed, the connector obtains the status of the ticket and update it in EIC.
9. The tasks in EIC are completed.