SERVICENOW

slack

REVANTH KARRA

ServiceNow Integrations|ITOM

revanth-karra

karrarevanth325@gmail.com

Hyderabad, India

Slack Integration with ServiceNow

Why we need to Integrate Slack with ServiceNow?

One of the biggest benefits of integrating Slack with ServiceNow is that it can help to increase efficiency and productivity. By bringing these two tools together, users can access and manage ServiceNow records directly from within Slack, without having to switch between applications.

Improved customer satisfaction

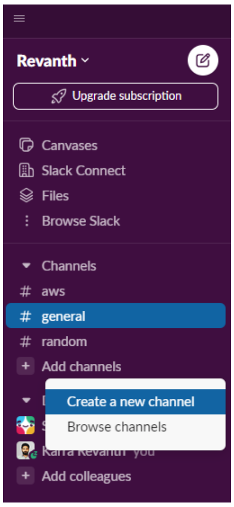
By integrating Slack with ServiceNow, organizations can provide their customers with a more seamless and efficient support experience. Customers can submit and track requests directly from within Slack, and receive updates on the status of their requests in real time. This can help to improve customer satisfaction and loyalty.

Examples:

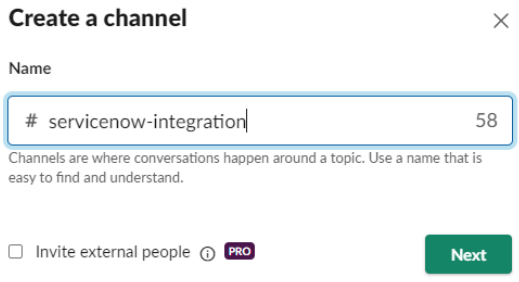
- Create and manage ServiceNow incidents directly from Slack.
- Receive notifications about new or updated ServiceNow records in Slack.
- Search for and view ServiceNow records in Slack.
- Automate tasks between Slack and ServiceNow.
- Collaborate with other users on ServiceNow records in Slack.

Let’s get started!

- Open the slack and click on Add channel and select create a new channel.



- Give a name to our channel (Example: servicenow-integration).


Create a channel

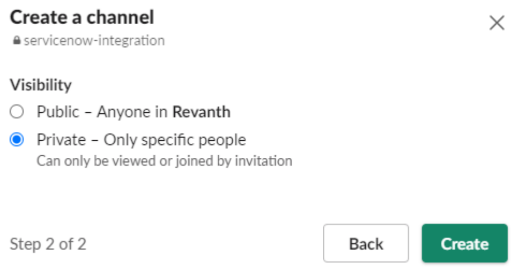
Name

servicenow-integration 58

Channels are where conversations happen around a topic. Use a name that is easy to find and understand.

□ Invite external people PRO

- For now, I’ll set to private for this demo.


Create a channel

servicenow-integration

Visibility

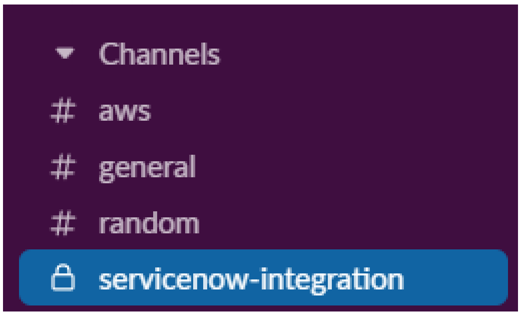
○ Public – Anyone in **Revanth**

● Private – Only specific people

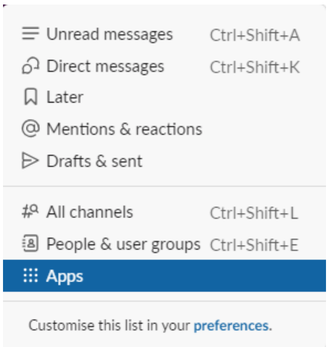
Can only be viewed or joined by invitation

Step 2 of 2

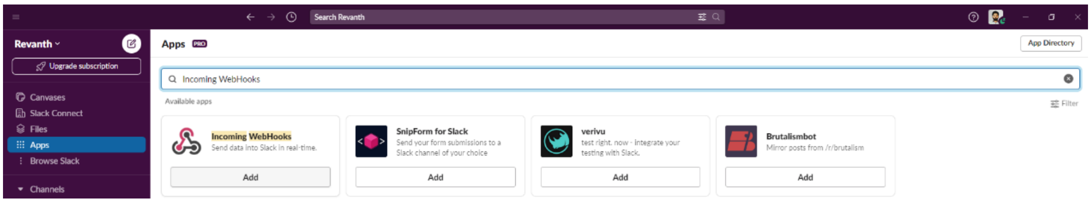
- Now a new channel has been created under our Channel module.



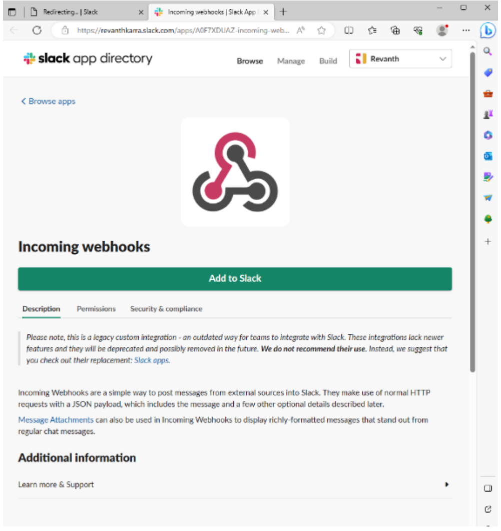
- Now, let us go to the option called Browse Slack > Apps.


Unread messages Ctrl+Shift+A
Direct messages Ctrl+Shift+K
Later
Mentions & reactions
Drafts & sent
All channels Ctrl+Shift+L
People & user groups Ctrl+Shift+E
Apps
Customise this list in your preferences.

- And search for Incoming WebHooks and hit the Add option.



- Under Incoming WebHooks, click on Add to Slack option.



Incoming webhooks

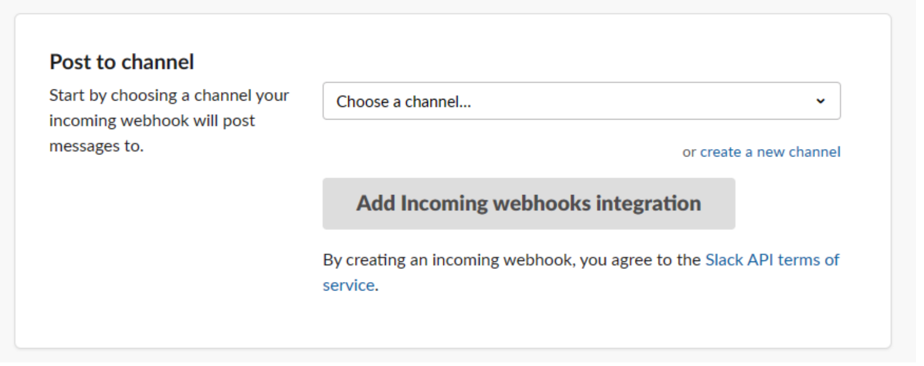
Please note, this is a legacy custom integration - an outdated way for teams to integrate with Slack. These integrations lack newer features and they will be deprecated and possibly removed in the future. ***We do not recommend their use.*** *Instead, we suggest that you check out their replacement: Slack apps.*

Incoming Webhooks are a simple way to post messages from external sources into Slack. They make use of normal HTTP requests with a JSON payload, which includes the message and a few other optional details described later.

Message Attachments can also be used in Incoming Webhooks to display richly-formatted messages that stand out from regular chat messages.

Additional information

- Let us choose our created channel in slack and hit Add Incoming Webhooks Integration button.

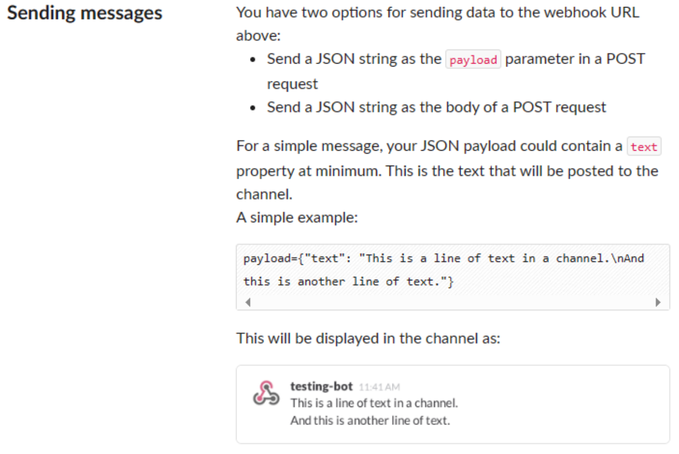


Post to channel

Start by choosing a channel your incoming webhook will post messages to.

By creating an incoming webhook, you agree to the Slack API terms of service.

- Scroll down and click copy WebHook URL.
- WebHook URL format be will like this https://hooks.slack.com/services/123ABCXYZ
- Here, we need to remember the format of the payload under Sending messages, in order to apply in our HTTP method content body.



Sending messages

You have two options for sending data to the webhook URL above:

- Send a JSON string as the `payload` parameter in a POST request
- Send a JSON string as the body of a POST request

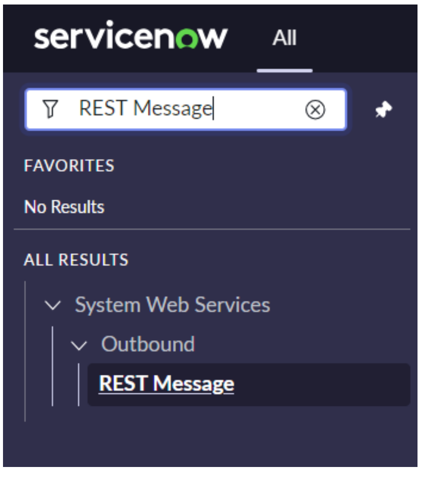
For a simple message, your JSON payload could contain a `text` property at minimum. This is the text that will be posted to the channel.
A simple example:

```
payload={"text": "This is a line of text in a channel.\nAnd this is another line of text."}
```

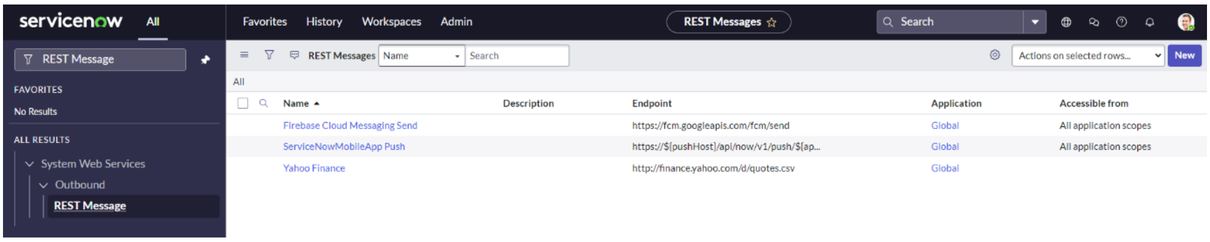
This will be displayed in the channel as:

testing-bot 11:41 AM
This is a line of text in a channel.
And this is another line of text.

- Next go to ServiceNow PDI, And search for REST Message.
- Path: System Web Services > Outbound > REST Message.

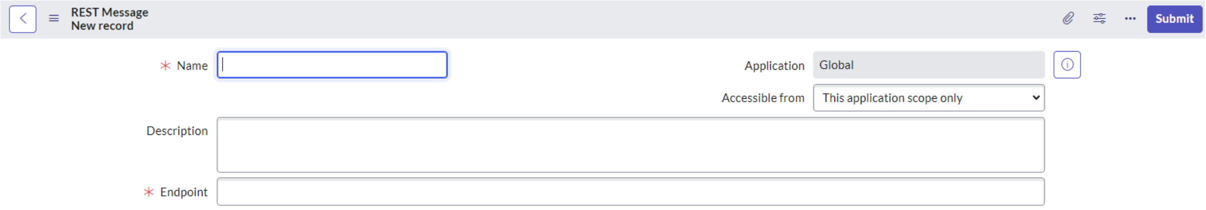


- Click New on the REST Message header.



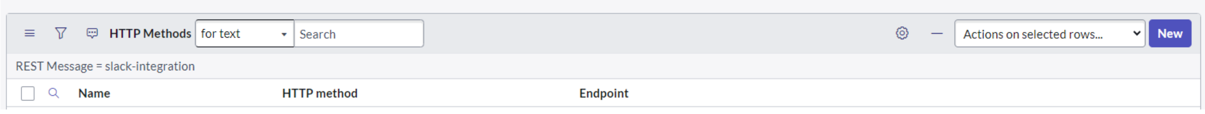
Name ▲	Description	Endpoint	Application	Accessible from
Firebase Cloud Messaging Send		https://fcm.googleapis.com/fcm/send	Global	All application scopes
ServiceNowMobileApp Push		https://${pushHost}/api/now/v1/push/${ap...	Global	All application scopes
Yahoo Finance		http://finance.yahoo.com/d/quotes.csv	Global	

- You will be re-directed to REST Message New Record.



* Name	
Application	Global
Accessible from	This application scope only
Description	
* Endpoint	

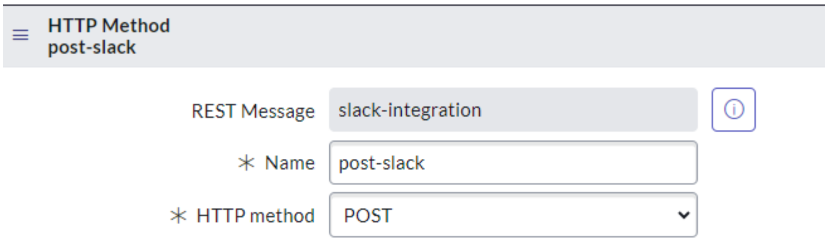
- And give the name for our REST Message Record (Example: slack-integration)
- We have to paste the copied WebHook URL in the Endpoint field.
- Right click on the REST Message slack-integration record header and click Save.
- Now we can find HTTP Methods.



REST Message = slack-integration

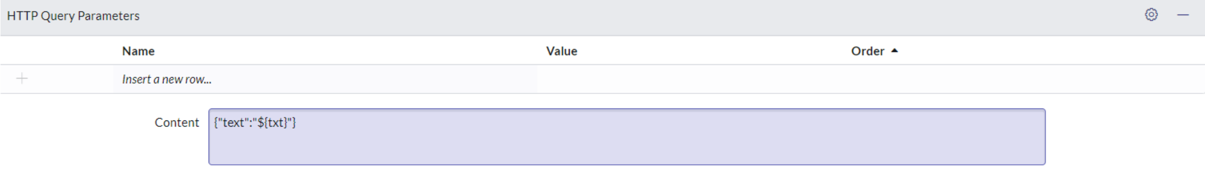
Name	HTTP method	Endpoint

- We can directly Edit our record in HTTP Methods table by clicking Default GET option under Name.
- Let us now change the name and HTTP Method.



REST Message	slack-integration
* Name	post-slack
* HTTP method	POST

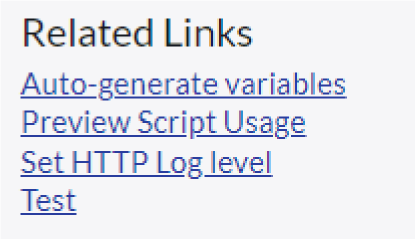
- Now let’s click on HTTP Request.
- Under HTTP Query Parameters > Content, we need to provide the content field in this payload format {"text":"${txt}"}



Name	Value	Order ▲
Insert a new row...		

Content {"text":"${txt}"}

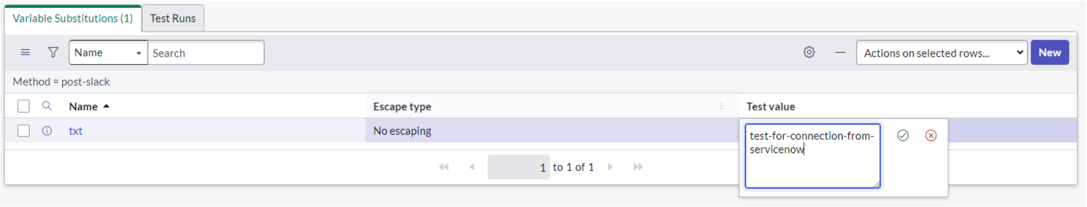
- Now right click on the HTTP Method post-slack header and hit Save.
- Under related links, Click on Auto-generate variables



Related Links

Auto-generate variables
Preview Script Usage
Set HTTP Log level
Test

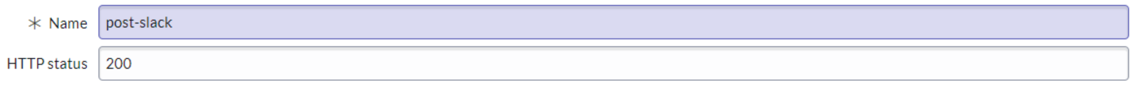
- Now we can see variable substitutions table is autogenerated with the variable as mentioned in the content field.
- Let us edit by double clicking on the empty field under Test value and give the naming convention (Example: test-for-connection-from-servicenow) and click Save.



Method = post-slack

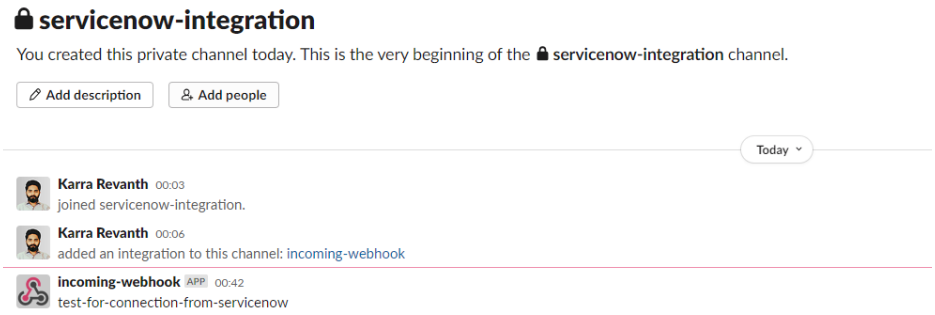
Name	Escape type	Test value
txt	No escaping	test-for-connection-from-servicenow

- Now click on Test under Related Links.
- We will get the HTTP status as 200 for our post-slack record.



✱ Name	post-slack
HTTP status	200

- Now if we go to our channel servicenow-integration in slack, we can find the test-for-connection-from-servicenow message.



servicenow-integration

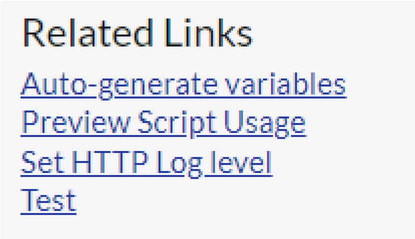
You created this private channel today. This is the very beginning of the **servicenow-integration** channel.

Karra Revanth 00:03
joined servicenow-integration.

Karra Revanth 00:06
added an integration to this channel: incoming-webhook

incoming-webhook APP 00:42
test-for-connection-from-servicenow

- We successfully Integrated servicenow with slack.
- In order to show the dynamic use case scenario, I am using problem table from now on.
- If we want to get all the updates regarding the records on the problem table in our slack channel dynamically, we need to create a business rule in the problem table.
- Now let us click on Preview Script Usage under Related Links for post-slack method.



Related Links

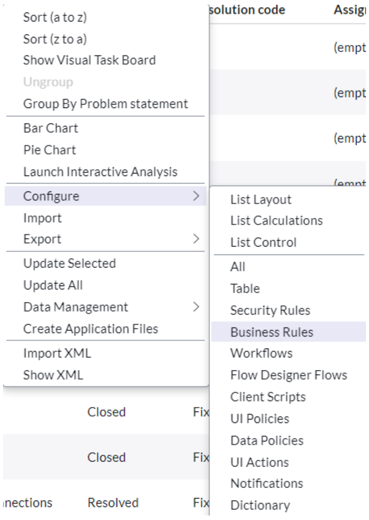
Auto-generate variables

Preview Script Usage

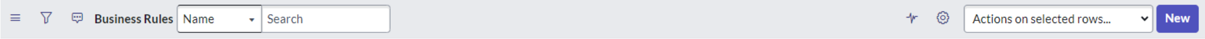
Set HTTP Log level

Test

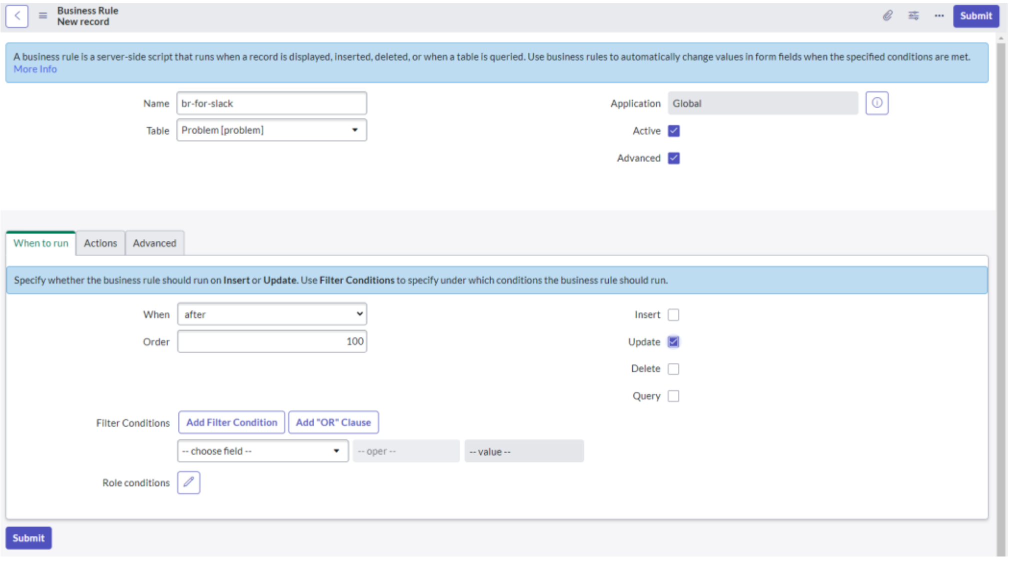
- Now we will get a popup regarding Preview REST Message script usage and copy the entire script.
- Now let us go to problem table and configure a business rule by clicking problem.list in the navigation.
- By right clicking on the Fields, we will get different option. Here we need to choose configure > business rules.



- We will be redirected to Business Rules table, here we need to click on New.



- Now Let us give a name for our new Business Rule record (Example: br-for-slack) and make sure to select the check box for advance option
- For this demo let us choose when to run condition as after update. For doing that make sure to select after option for When condition and select Update check box.



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	br-for-slack	Application	Global
Table	Problem [problem]	Active	☑
		Advanced	☑

Specify whether the business rule should run on **Insert** or **Update**. Use **Filter Conditions** to specify under which conditions the business rule should run.

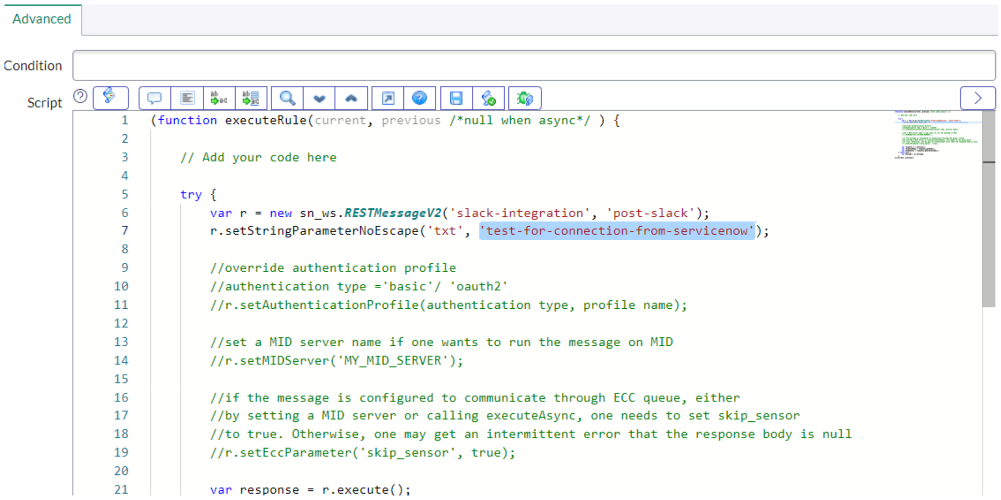
When	after	Insert	□
Order	100	Update	☑
		Delete	□
		Query	□

Filter Conditions: Add Filter Condition | Add "OR" Clause

-- choose field -- | -- oper -- | -- value --

Role conditions

- Now let us go to advanced, in this we have to paste the copied script from Preview Script Usage
- Replace this line (test-for-connection-from-servicenow) in the script to make it dynamic. So, that it will send the problem number, problem statement and CI name to slack for every update on the problem record.

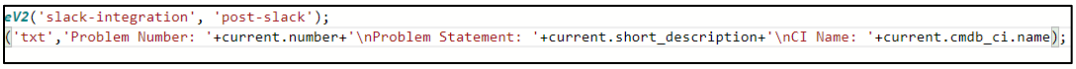


```
(function executeRule(current, previous /*null when async*/ ) {

    // Add your code here

    try {
        var r = new sn_ws.RESTMessageV2('slack-integration', 'post-slack');
        r.setStringParameterNoEscape('txt', 'test-for-connection-from-servicenow');

        //override authentication profile
        //authentication type ='basic'/ 'oauth2'
        //r.setAuthenticationProfile(authentication type, profile name);

        //set a MID server name if one wants to run the message on MID
        //r.setMIDServer('MY_MID_SERVER');

        //if the message is configured to communicate through ECC queue, either
        //by setting a MID server or calling executeAsync, one needs to set skip_sensor
        //to true. Otherwise, one may get an intermittent error that the response body is null
        //r.setEccParameter('skip_sensor', true);

        var response = r.execute();
```

- Make sure to edit the line as below to make it dynamic and also check for exact backend names as per your requirement.

  Script:

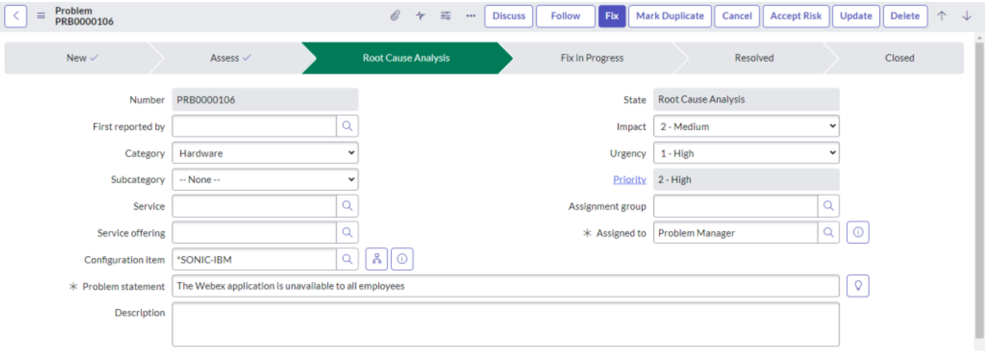
  r.setStringParameterNoEscape('txt','Problem Number: '+current.number+'\nProblem Statement: '+current.short_description+'\nCI Name: '+current.cmdb_ci.name);



```
eV2('slack-integration', 'post-slack');
('txt','Problem Number: '+current.number+'\nProblem Statement: '+current.short_description+'\nCI Name: '+current.cmdb_ci.name);
```

- Right click on the header and select Save.

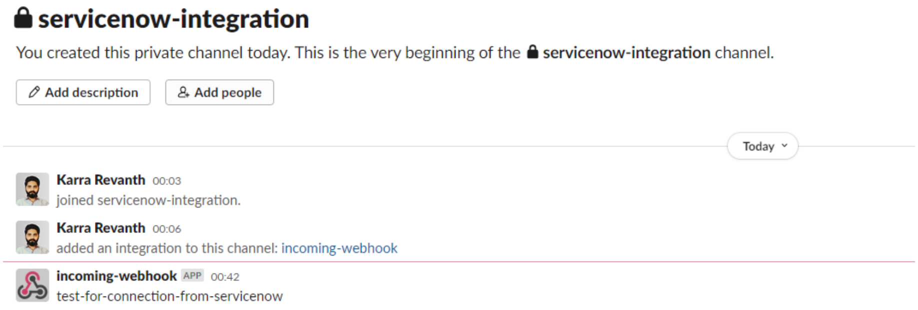
- Now let us go to our problem table(problem.list) and test .
- From the table, select any random record.
- For this practical demo, Now let us update the configuration item as *SONIC-IBM



Number	PRB0000106	State	Root Cause Analysis
First reported by		Impact	2 - Medium
Category	Hardware	Urgency	1 - High
Subcategory	-- None --	Priority	2 - High
Service		Assignment group	
Service offering		✱ Assigned to	Problem Manager
Configuration item	*SONIC-IBM		
✱ Problem statement	The Webex application is unavailable to all employees		
Description			

- Right click on the header and click save. So that, it will be considered as update.
- Now let us go to slack and check for the updated problem record details.
- So, if someone make changes to any record in the problem table, we will get an update to our slack channel because we already know that we set the business rule condition as After Update.
- We successfully received the problem details in our slack channel with problem number, problem statement and CI Name

BEFORE:



servicenow-integration

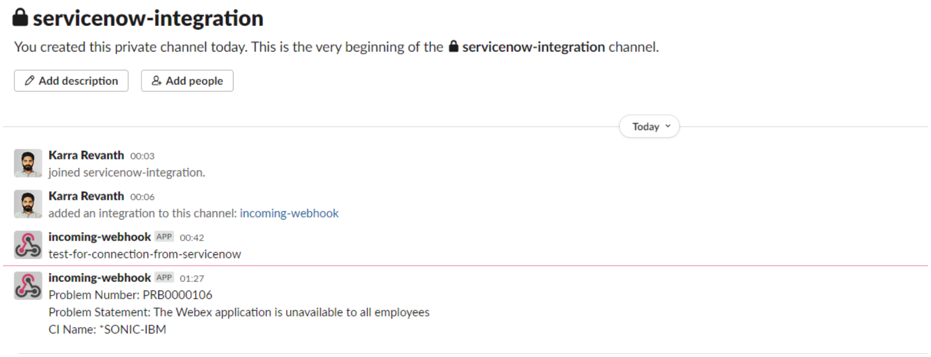
You created this private channel today. This is the very beginning of the **servicenow-integration** channel.

Karra Revanth 00:03
joined servicenow-integration.

Karra Revanth 00:06
added an integration to this channel: incoming-webhook

incoming-webhook APP 00:42
test-for-connection-from-servicenow

AFTER:



servicenow-integration

You created this private channel today. This is the very beginning of the **servicenow-integration** channel.

Karra Revanth 00:03
joined servicenow-integration.

Karra Revanth 00:06
added an integration to this channel: incoming-webhook

incoming-webhook APP 00:42
test-for-connection-from-servicenow

incoming-webhook APP 01:27
Problem Number: PRB0000106
Problem Statement: The Webex application is unavailable to all employees
CI Name: *SONIC-IBM