Slack Integration

Introduction

It is possible to push events generated via opEvents to external communication and collaboration tools, such as Slack using a Perl script.

Installation

Install the script

As a pre-requisite, you will need to create a scripts directory located within /usr/local/nmis9 (or where your NMIS9 installation resides), by executing the following command:

```
mkdir /usr/local/nmis9/scripts
```

Once complete, then upload the following script to the scripts folder you just created:

slack_messages.pl

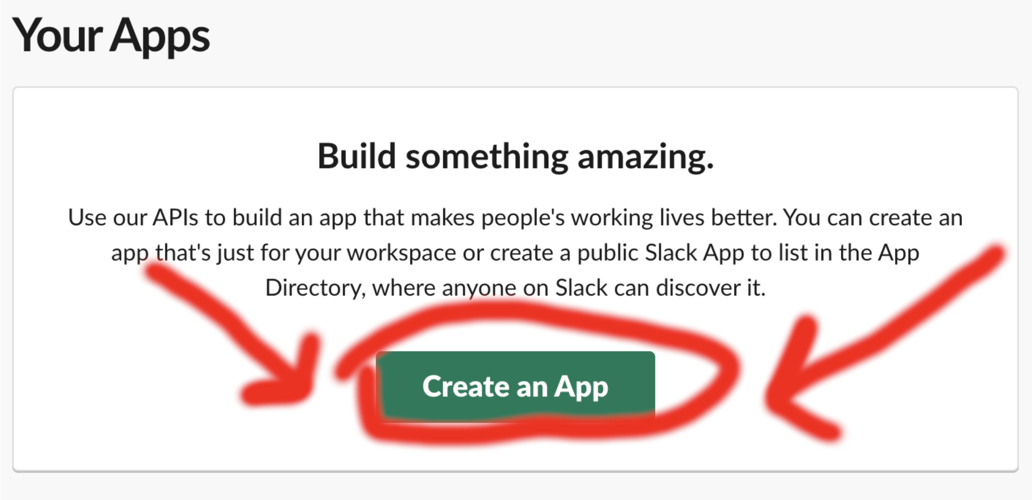
Set the script's permissions with the command:

```
/usr/local/nmis9/bin/nmis-cli act=fixperms
```

where /usr/local/nmis9 is the path to your NMIS installation.

Create your webhook

Navigate to the slack app page at https://api.slack.com/apps and click the "Create an App" button.

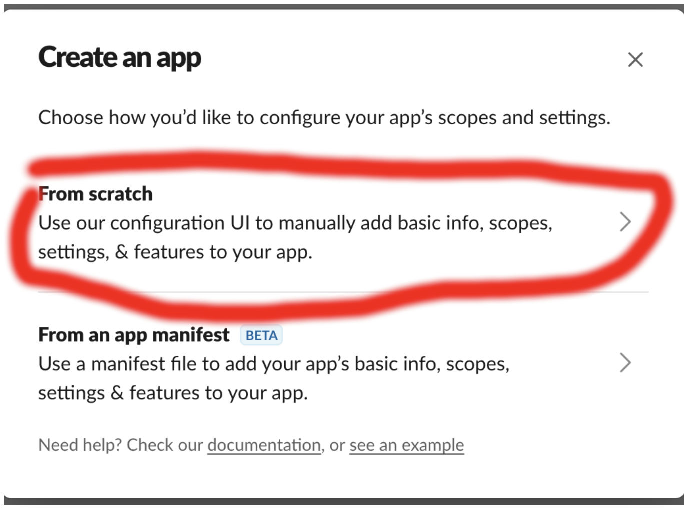


Your Apps

Build something amazing.

Use our APIs to build an app that makes people's working lives better. You can create an app that's just for your workspace or create a public Slack App to list in the App Directory, where anyone on Slack can discover it.

Select "From scratch":



Create an app

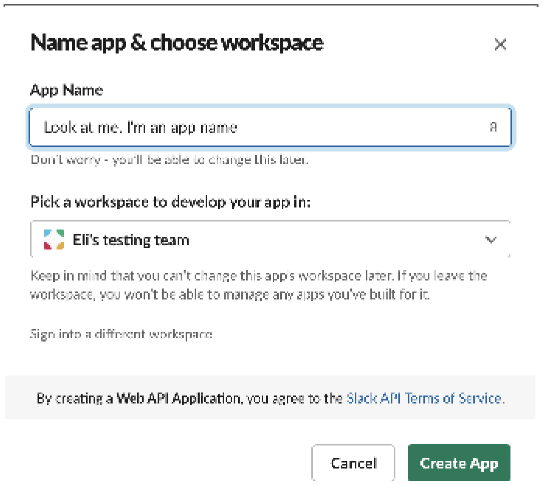
Choose how you'd like to configure your app's scopes and settings.

From scratch
Use our configuration UI to manually add basic info, scopes, settings, & features to your app.

From an app manifest BETA
Use a manifest file to add your app's basic info, scopes, settings & features to your app.

Need help? Check our documentation, or see an example

Enter a name for your app and pick the workspace you want to use. Click "Create App" when you are ready.



Name app & choose workspace

App Name

Look at me. I'm an app name

Don't worry - you'll be able to change this later.

Pick a workspace to develop your app in:

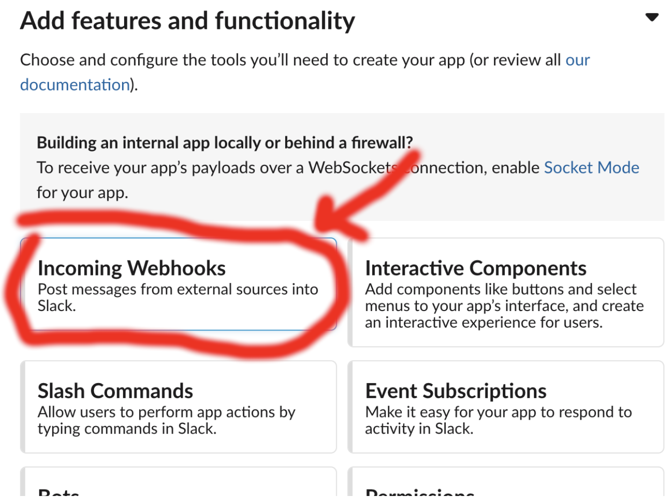
Eli's testing team

Keep in mind that you can't change this app's workspace later. If you leave the workspace, you won't be able to manage any apps you've built for it.

Sign into a different workspace

By creating a Web API Application, you agree to the Slack API Terms of Service.

Click "Incoming Webhooks":



Add features and functionality

Choose and configure the tools you'll need to create your app (or review all our documentation).

Building an internal app locally or behind a firewall?
To receive your app's payloads over a WebSockets connection, enable Socket Mode for your app.

Incoming Webhooks
Post messages from external sources into Slack.

Interactive Components
Add components like buttons and select menus to your app's interface, and create an interactive experience for users.

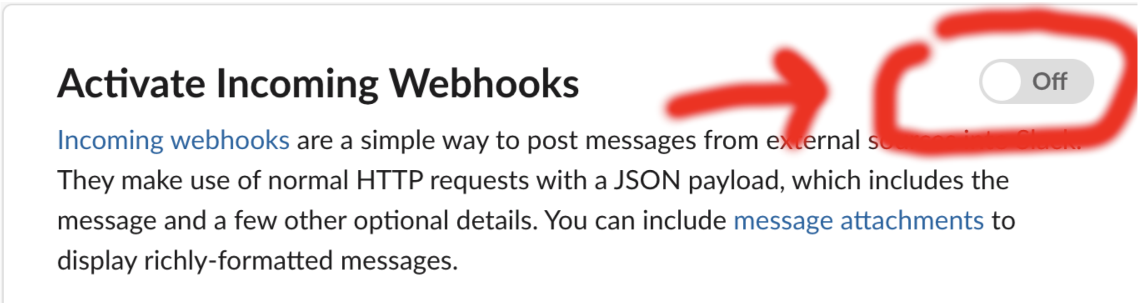
Slash Commands
Allow users to perform app actions by typing commands in Slack.

Event Subscriptions
Make it easy for your app to respond to activity in Slack.

Bots

Permissions

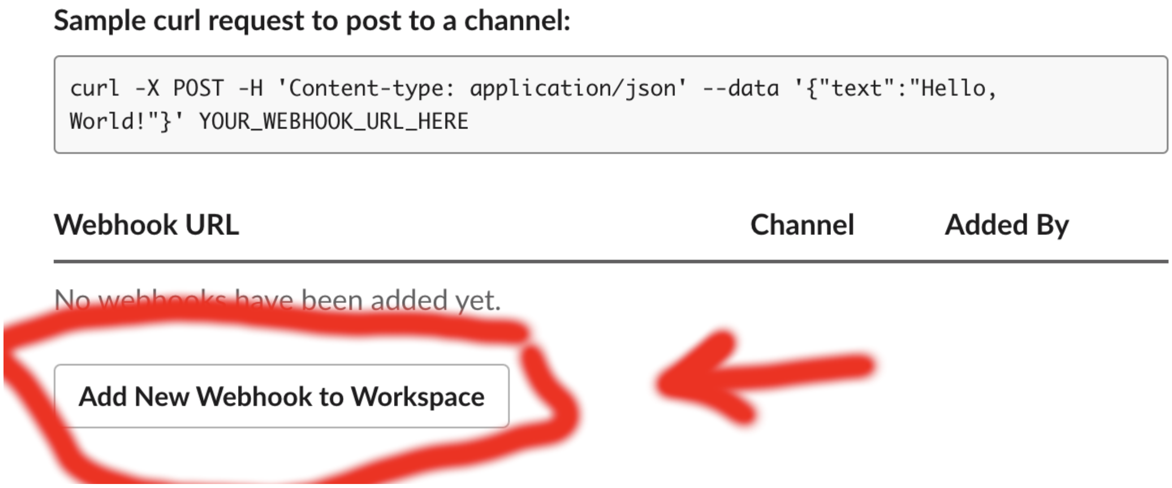
Click the switch to "Activate Incoming Webhooks":



Activate Incoming Webhooks

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. You can include message attachments to display richly-formatted messages.

Click "Add New Webhook to Workspace" at the bottom of the page:

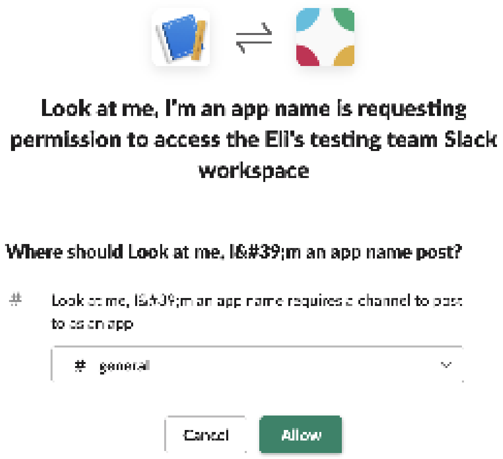


Sample curl request to post to a channel:

```
curl -X POST -H 'Content-type: application/json' --data '{"text":"Hello,
World!"}' YOUR_WEBHOOK_URL_HERE
```

Webhook URL	Channel	Added By
No webhooks have been added yet.		

Select the channel you want opEvents to post to and click allow to grant permission:

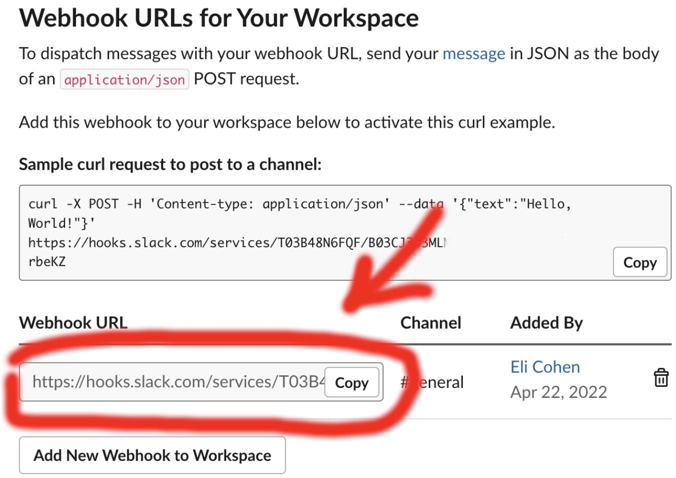


Look at me, I'm an app name is requesting permission to access the Eli's testing team Slack workspace

Where should Look at me, I&#39;m an app name post?

Look at me, I&#39;m an app name requires a channel to post to as an app

Scroll down to the bottom of the page and copy your webhook URL - you will need it for the next step.



Webhook URLs for Your Workspace

To dispatch messages with your webhook URL, send your message in JSON as the body of an `application/json` POST request.

Add this webhook to your workspace below to activate this curl example.

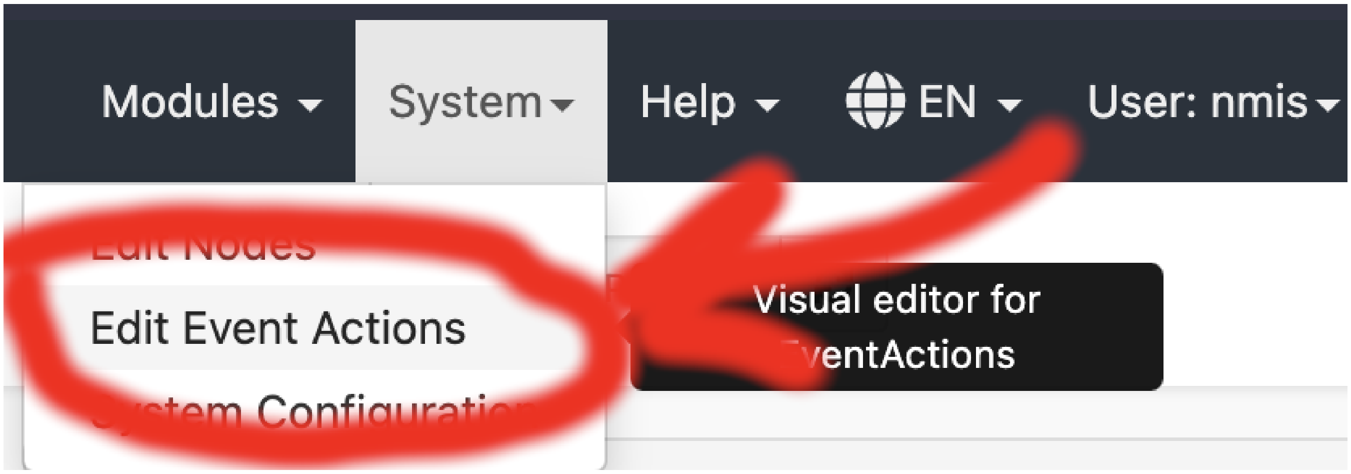
Sample curl request to post to a channel:

```
curl -X POST -H 'Content-type: application/json' --data '{"text":"Hello,
World!"}'
https://hooks.slack.com/services/T03B48N6FQF/B03CJ[illegible]3ML
rbeKZ
```

Webhook URL	Channel	Added By
https://hooks.slack.com/services/T03B4	# eneral	Eli Cohen Apr 22, 2022

Integrate with opEvents

Log into your opEvents installation from your browser. Once authenticated, in the upper right, click System, then Edit Event Actions.



Immediately under the line that reads:

```
"script" : {
```

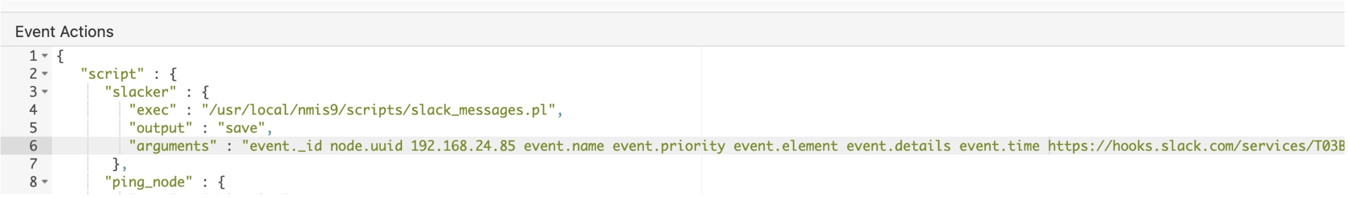
Add the text:

```
      "slacker" : {
         "exec" : "/usr/local/nmis9/scripts/slack_messages.pl",
         "output" : "save",
         "arguments" : "event._id node.uuid SERVER event.name event.priority event.element event.details event.
time WEBHOOK"
      }
```

Where:

- /usr/local/nmis9 is replaced with your NMIS path.
- SERVER is replaced with the IP or domain of your server (eg. 192.168.24.85, or nmis-01.mydomain.net).
- WEBHOOK is replaced with the webhook we copied in the last step.

When complete this should look like the following:



```
{
   "script" : {
      "slacker" : {
         "exec" : "/usr/local/nmis9/scripts/slack_messages.pl",
         "output" : "save",
         "arguments" : "event._id node.uuid 192.168.24.85 event.name event.priority event.element event.details event.time https://hooks.slack.com/services/T03B
      },
      "ping_node" : {
```

Then click the Validate button. If you see a 'Syntax OK' prompt, proceed to click Save. You can also click Save and this will also check the syntax is OK.

Add calls to the script

You can now call the script through opEvents with the command:

```
script.slacker()
```

For example, let's add a call to the script when a node goes down. This would look like the following:

```
            "30" : {
               "BREAK" : "false",
               "IF" : "event.event eq \"Node Down\"",
               "THEN" : [
                  "tag.isbroken(nodedown)",
                  "tag.verybad(42)",
                  "script.slacker()"
                ]
                        }
```

After modifying the Event Actions to suit your needs, you can then proceed to press the Save button, and verify the Syntax is OK. Once confirmed, you will then need to restart the opEvents daemon (opeventsd).

Testing the integration

If you have applied the integration to an event that may not occur on a regular occurrence, you can run the following command to simulate the event and verify your integration works:

```
/usr/local/omk/bin/opevents-cli.pl act=create-event node=DC-RTR-01 state=down event="Node Down" stateful="Node"
```

Troubleshooting

exec failed: Permission denied

In the Event Context of a Event where you have included the script, you see the following in the Scripts section under **slacker**:

```
exec failed: Permission denied
```

To resolve this, you need to make the following modification to the "exec" section:

```
     "slacker" : {
        "exec" : "perl /usr/local/nmis9/scripts/slack_messages.pl",
        "output" : "save",
        "arguments" : "event._id node.uuid SERVER event.name event.priority event.element event.details event.
time WEBHOOK"
       }
```

You will need to restart the opEvents daemon after making this change.

Slack push notifications - This content can't be displayed.

You may find that Slack push notifications display a "This content can't be displayed" message on some operating systems. A quick modification to your slack_messages.pl will resolve this. Locate the following lines:

```
               \"username\": \"$server opEvents Notifier\",\n
```

Then underneath both of these (there should be two lines), add the following. Below is an example, but you can modify as you see necessary. Ensure to verify your formatting once changes have been made.

```
                                 \"text\": \"$node - $event - Priority: $prioritynum\",
```