|  |
| --- |
| Events/Notifications  Slack + Webhooks + Java |

Webhooks

Slack, employing **Webhooks** technology, can communicate with other Slack applications within the same organization/computer installation registered to the (cloud-based) **Slack Workspace**.

The same **Webhooks** technology enables a **Slack Workspace** to communicate with applications running at an external computer installation.

For instance, an external computer installation can POST/cURL-to an **Incoming Slack Application.**

While Slack is **OAuth2**-capable, this security feature is out of scope for this discussion.





**Part One** covers creating a proprietary Slack token, capable of acting as a switching mechanism for a passive Receiver/Destination for incoming messages.

Begin by following the instructions found at this link:

ref <https://api.slack.com/tutorials/slack-apps-hello-world>

This document illustrates an example Slack session that creates a Slack Application which acts as what Slack call an **Incoming Webhook**.

If you have questions while following the excellent instructions on the Slack web site (at the above link), maybe my experience in creating a Slack Application, captured in this document, will provide clarity.





**Part Two** covers the Java code that acts as a Publisher of messages to what Slack calls a **Slack Application**, which consists of client code which forwards messages to a Slack (communal) Workspace.

In between the Java call site and the (Slack-fabricated) **Slack Application**, the proprietary Slack token is used to route messages to another endpoint, using either an **IRC gateway** or an **XMPP gateway**.

The gateway comprises the Network layer (**OSI Layer 3**). Firewall-tunneling may be applicable in some circumstances.

The exact nature of Slack’s underlying transmission Data Link (**OSI Layer 2**) protocol is not known.



Part One

This section presents the screen shots from a succession of steps that create a **Slack Webhook Token**, and a **Slack Application**, which is associated with an extant Slack Workspace (**filolifo55555**).

The screen shots were culled from me following along with the instructions contained in the page shown below – from the link:

<https://api.slack.com/tutorials/slack-apps-hello-world>:

Graphical user interface, application

Description automatically generated

Notice the above screen shot has text: **app creation page**, which actually is a hyperlink:

<https://api.slack.com/apps/new>

This link launches a page that is titled **Create a Slack App** (next page).

Graphical user interface, text, application, email

Description automatically generated

Notice above, that the **Pilot** Slack App is created within the **filolifo55555** Development Slack Workspace.

|  |  |  |
| --- | --- | --- |
| Click the | A close up of a sign  Description automatically generated | button. |



**NB** Before creating any (of the proffered types of) **Slack Applications**, be sure to record the **Webhooks Credentials** that you will need later,

Unfortunately, the **Webhooks Credentials** appear after the **Slack Applications** options, and choosing any option navigates you away from the very important **Webhooks Credentials.**

IMHO: Despite the caveats on the page, there should be an export option



The crucial information depicted below appears after the menu options that when elected, navigates-away from the **App Credentials** page.

To view (*and record*) the information shown next, scroll.

Text

Description automatically generated

**NB** Do not make any menu/options selection on the **Building Apps for Slack** landing page (depicted immediately below) until you view (*and record*) the **App Credentials** information (as shown above, on the previous page).

Since a copy/paste of the **App Credentials** page’s content using Word does not retain the values (just their labels) – use something like Atom instead to retain the important credentials.

*On the Slack website*, the **App Credentials** page’s information is presented below the **Building Apps for Slack** (landing page, arrived-at from the **Create a Slack App** page).

Making a selection on the **Building Apps for Slack** takes you away from the crucial **App Credentials** page’s information.

So, scroll.



Text, table

Description automatically generated

Double-click the **Incoming Webhooks** option, which launches the **Incoming Webhooks** page (see next page).



Text, letter

Description automatically generated

|  |  |
| --- | --- |
| NB Assure the **On** state of the slider in the upper right-hand corner of the above page: | Graphical user interface, application, chat or text message  Description automatically generated |



Graphical user interface, text, application

Description automatically generated



|  |
| --- |
| Graphical user interface, text, application, website  Description automatically generated |

Click  **Alloww**



Graphical user interface, text, application

Description automatically generated

You will be returned to the above page, where you can copy a test link:

<https://hooks.slack.com/services/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>

Leveraging cURL, append the above URI to a to the proprietary Slack URI to test the viability of the network connectivity for the **Incoming Webhook** that you just created.

<https://hooks.slack.com/services/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>

Graphical user interface, text, application

Description automatically generated

The **Your Apps** page, shown above already contains a Slack app named **gregmarty**. This app was previously created to interact with the same **Slack Workspace**, which is named **filolifo55555**.

In that Slack Workspace, shown later, you’ll see messages dating back to July 1, 2018 (when the Slack Workspace and the **Slack App** were each created together).

The new **Guide** **Slack Application** will interact with that same **Slack Workspace** (**filolifo55555**), by means of the Slack Webhook.

Notice the green Create New App button (upper-right corner).

|  |  |
| --- | --- |
| Click the green Create New App button (upper-right corner) | A close up of a sign  Description automatically generated |



The balance of the steps outlined in this section (**Part One)** show how an extant Slack Workspace is associated with a newly-minted Slack App, and how it can be tested using a cURL command that is “wired” to a Java Publisher (client code).

The second part of this document outlines the Java participant in the Slack App’s Webhook contract that fulfills the role of Publisher.



**Incoming Webhook: Testing the Functionality**

Validate the **Incoming Webhook App** by sending a few cURL calls from a terminal, observing the below format/convention (which incorporates the Webhook w/ the Slack URI).



**Call One**

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

<https://hooks.slack.com/services/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>

Text

Description automatically generated





**Call Two**

curl -X POST -H 'Content-type: application/json' --data '{"text":"Greetings…"}' \

<https://hooks.slack.com/services/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>

Text

Description automatically generated



**Slack Application’s Incoming/Receptor Response**

**A screen shot of a computer

Description automatically generated**



**Uninstalling a Webhook**

Follow the below instructions, using this link:

<https://filolifo55555.slack.com/apps>

**Graphical user interface, text, application

Description automatically generated**



Part Two

The **Webhooks** technology delivered by **Slack**, as a **Slack Incoming Application,** allows that **App** – within a **Stack Workspace** – to act as an **HTTP endpoint** for calling code.

In our case, we want a Java HTTP client to emit messages to the **Stack App/Webhook**.



A quick test to prove the **Stack Workspace** is being sent messages can be done using cURL.

The CLI modality of call can serve as a baseline test for efficacy of the Java client we are writing.

The two complementary types of **Slack**-centric call sites are covered next.



CLI / cURL

The cURL call-site command

A picture containing text

Description automatically generated

A screenshot of a cell phone

Description automatically generated

The Slack (the recipient’s) Result



Java HTTP Client

The Java call-site command

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

The Slack (the recipient’s) Result



Code / Messaging

Which is “better”?

Text

Description automatically generated



Code / Messaging

The Envoy

Text

Description automatically generated



Code / Messaging

The HTTP Solicitor (to Slack)

Text

Description automatically generated



Lifecycle Scripting (no IDE)

Graphical user interface, text

Description automatically generated

