Slack Integration

- Introduction
    - Features
    - Prepare your Inbenta instances
        - Create translations object in ExtraInfo (optional)
        - HyperChat integration (optional)
    - Building the Slack Connector
        - Setup Chatbot Slack Connector
            - Required Configuration
            - Optional Configuration
        - HYPERCHAT (chat.php)
        - CONVERSATION (conversation.php)
        - ENVIRONMENTS (environments.php)
        - Deployment
- Slack Application creation
- Authorization & Permissions
- Event subscriptions
- Interactive Components
- Installing App on the Workspace

Introduction

The purpose of this document is to provide the details of the integration of Inbenta's Chatbot and Hyperchat solution with Slack.

Features

These are the supported answer types and features:

- FAQ Intents with images, videos and Hyperlinks as external links.
- Related Contents.
- Sidebubble content (will show up in the main chat window).

- Multiple options.
- Polar Questions.
- Dialogs.
- Forms, Actions & Variables.
- Hyperchat Integration.
- Ratings.

Prepare your Inbenta instances

Create translations object in ExtraInfo (optional)

You can manage the translation labels from Extra Info. Here are the steps to create the translations object:

1. In your Backstage instance, go to Knowledge Extra Info and click on 'Manage groups and types' slack Add type. Name it '**translations**' and add a new property with type 'Multiple' named with your chatbot's language label (en, es, it...).
2. Inside the language object, add all the labels that you want to override. Each label should be a 'text' type entry (you can find the labels list below).
3. Save your translations object.

Now you can create the ExtraInfo object by clicking the **New entry** button, selecting the 'translations' type and naming it as 'translations'. Then, fill each label with your desired translation and remember to publish ExtraInfo by clicking the **Post** button.

Here you have the current labels with their English value:

- **agent_joined** => 'Agent $agentName has joined the conversation.'
- **api_timeout** => 'Please, reformulate your question.'
- **ask_rating_comment** => 'Please tell us why'
- **ask_to_escalate** => 'Do you want to start a chat with a human agent?'
- **chat_closed** => 'Chat closed'
- **creating_chat** => 'I will try to connect you with an agent. Please wait.'
- **error_creating_chat** => 'There was an error joining the chat'
- **escalation_rejected** => 'What else can I do for you?'

- **no** => 'No'
- **no_agents** => 'No agents available'
- **queue_estimation_first** => ‘There is one person ahead of you.’
- **queue_estimation** => ‘There are $queuePosition people ahead of you.’
- **rate_content_intro** => 'Was this answer helpful?'
- **thanks** => 'Thanks!'
- **yes** => 'Yes'
- **ticket_response_intro** => "Hi, I'm the agent that you spoke to a while ago. My response to your question"
- **ticket_response_info** => "Here is the ticket number for your reference"
- **ticket_response_end** => "You can now continue chatting with the chatbot. If you want to talk to someone, type 'agent'. Thank you!"

Information
Remember to publish your ExtraInfo changes by clicking the ‘Post’ button.

Even if you have created the ExtraInfo translation, it is mandatory that the lang file exists in the “lang” folder.

HyperChat integration (optional)

If you use HyperChat you must subscribe your UI to the Hyperchat events. Open your *Messenger* instance. Go to Messenger Settings Chat Webhooks. Here, in the ‘Events’ column type “queues:update,invitations:new,invitations:accept,forever:alone,chats:close, messages:new, users:activity”. In the ‘Target’ column paste your UI’s URL, then click on the ‘+’ button on the right.

Building the Slack Connector

Setup Chatbot Slack Connector

Required Configuration

In your UI directory, go to **conf**. Here, you have a readme file with some structure and usage explanations. If you only want to build a chatbot, fill the **key** and **secret** values inside the **conf /custom/api.php** file with your Inbenta Chatbot API credentials. If you want to modify other configuration parameters, copy the desired file(s) from **conf/default** into **conf/custom** and modify the values.

Optional Configuration

There are some optional features that can be enabled from the configuration files. Every optional configuration file should be copied from **/conf/default** and store the custom version in **/conf /custom**. The bot will detect the customization and it will load the customized version. These are the optional configuration files and the configuration fields description.

HYPERCHAT (chat.php)

- **chat**
  - **enabled**: Enable or disable HyperChat ("**true**" or "**false**").
  - **version**: HyperChat version. The default and latest one is 1.
  - **appId**: The ID of the HyperChat app. This defines the instance in which the chat opens. You can find it in your instance Messenger Settings Chat.
  - **secret**: Your HyperChat instance application secret. You can find it in your instance Messenger Settings Chat.
  - **roomId**: The room where the chat opens. This is mapped directly to a Backstage queue ID. Numeric value, not a string. You can find your rooms list it in your instance Messenger Settings Queues.
  - **lang**: Language code (in ISO 639-1 format) for the current chat. This is used when the engine checks if there are agents available for this language to assign the chat to one of them.
  - **source**: Source id from the sources in your instance. Numeric value, not a string. The default value is **3 - Chat**. You can find your sources list it in your instance Messenger Settings Sources.
  - **regionServer**: The geographical region where the HyperChat app lives.
  - **server**: The Hyperchat server URL assigned to your instance. Ask your Inbenta contact for this configuration parameter.
  - **server_port**: The port where to communicate with the Hyperchat server. It's defined in your instance Messenger Settings Chat -->Port
  - **queue**:

        - **active**: Enable or disable the queue system (“**true**” or “**false**”). It **MUST** be enabled in your instance too (Messenger Settings Chat Queue mode).
- **triesBeforeEscalation**: Number of no-result answers in a row after the bot should escalate to an agent (if available). Numeric value, not a string. Zero means it’s disabled.
- **negativeRatingsBeforeEscalation**: Number of negative content ratings in a row after the bot should escalate to an agent (if available). Numeric value, not a string. Zero means it’s disabled.
- **messenger**: Setting that allows replying to tickets after the agent conversation is closed.
    - **auht_url**: Url for authorization, used by API Messenger.
    - **key**: API Key of the Messenger Instance. You can find it in Administration API [Production | Development].
    - **secret**: Secret Key token of the Messenger Instance. You can find it in Administration API [Production | Development].
    - **webhook_secret**: Secret token, defined when the configuration of External Ticket Source is made.

CONVERSATION (conversation.php)

- **default:** Contains the API conversation configuration. The values are described below:
    - **answers:**
        - **sideBubbleAttributes:** Dynamic settings to show side-bubble content. Because there is no side-bubble in Slack the content is shown after the main answer.
        - **answerAttributes:** Dynamic settings to show as the bot answer. The default is ["ANSWER_TEXT"]. Setting multiple dynamic settings generates a bot answer with concatenated values with a newline character (\n).
        - **maxOptions:** Maximum number of options returned in a multiple-choice answer.
    - **forms**
        - **allowUserToAbandonForm:** Whether or not a user is allowed to abandon the form after a number of consecutive failed answers. The default value is **true**.
        - **errorRetries:** The number of times a user can fail a form field before being asked if he wants to leave the form. The default value is 3.

    - **lang:** Language of the bot, represented by its ISO 639-1 code. Accepted values: ca, de, en, es, fr, it, ja, ko, nl, pt, zh, ru, ar, hu, eu, ro, gl, da, sv, no, tr, cs, fi, pl, el, th, id, uk
- **user_type**: Profile identifier from the Backstage knowledge base. Minimum:0. Default:0. You can find your profile list in your Chatbot Instance Settings User Types.
- **source**: Source identifier (e.g. "slack") used to filter the logs in the dashboards.
- **content_ratings**
    - **enabled**: Enable or disable the rating feature ("**true**" or "**false**").
    - **ratings**: Array of options to display in order to rate the content. Every option has the following parameters:
        - **id:** Id of your content rating. You can find your content ratings in your Chatbot instance Settings Ratings. Remember that your rating type should be "**content**".
        - **label:** Key of the label translation to display within the rating option button. The available labels can be configured from **/lang/**. Also can be modified from Backstage as described in section **Create translations object in ExtraInfo (optional)**
        - **comment:** If **true**, asks for a comment for the rating. It's useful when a user rates a content negatively in order to ask why the negative rating.
        - **isNegative:** If **true**, the bot will increment the negative-comments counter in order to escalate with an agent (if HyperChat **negativeRatingsBeforeEscalatio** is configured).

ENVIRONMENTS (environments.php)

This file allows configuring a rule to detect the current environment for the connector. It can check the current **http_host** or the **script_name** in order to detect the environment.

- **development:**
    - **type**: Detection type: check the **http_host** (e.g. *www.example.com*) or the **script_name** (e.g. */path/to/the/connector/server.php*).
    - **regex**: Regex to match with the detection type (e.g. "*/^dev.mydomain.com$/m*" will set the "development" environment when the detection type is *dev.example.com*).

- **preproduction**:
    - **type**: Detection type: check the **http_host** (e.g. *www.example.com*) or the **script_name** (e.g. */path/to/the/connector/server.php*).
    - **regex**: Regex to match with the detection type (e.g. “*/^.*/staging/.*$/m*“ will set the “preproduction” environment when the detection type contains “*/staging/*”).

Deployment

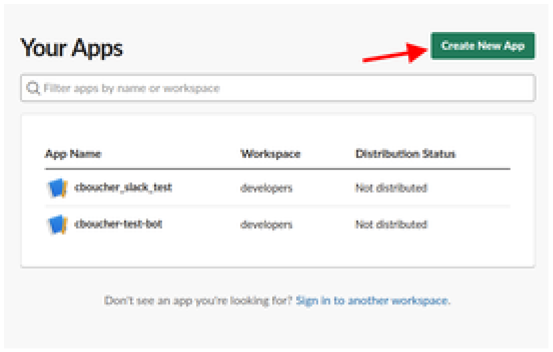
The Slack template must be served by a public web server in order to allow Slack to send the events to it. The environment where the template has been developed and tested has the following specifications

- Apache 2.4
- PHP 7.3
- PHP Curl extension
- Non-CPU-bound
- The latest version of **Composer** (Dependency Manager for PHP) to install all dependencies that Inbenta requires for the integration.
- If the client has a **distributed infrastructure**, this means that multiple servers can manage the user session, they must adapt their SessionHandler so that the entire session is shared among all its servers.

Slack Application creation

First is to go this url: https://api.slack.com/apps

Click on **Create New App**, then enter the name you want and select the **Workspace** on where you want to install the bot.

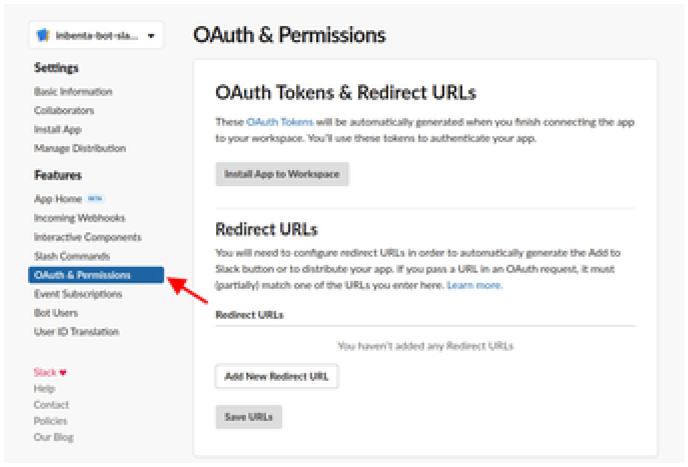


App Name	Workspace	Distribution Status
cboucher_slack_test	developers	Not distributed
cboucher-test-bot	developers	Not distributed

Don't see an app you're looking for? Sign in to another workspace.

Authorization & Permissions

On the left menu, click on **OAuth & Permissions**.



OAuth & Permissions

OAuth Tokens & Redirect URLs

These OAuth Tokens will be automatically generated when you finish connecting the app to your workspace. You'll use these tokens to authenticate your app.

Redirect URLs

You will need to configure redirect URLs in order to automatically generate the Add to Slack button or to distribute your app. If you pass a URL in an OAuth request, it must (partially) match one of the URLs you enter here. Learn more.

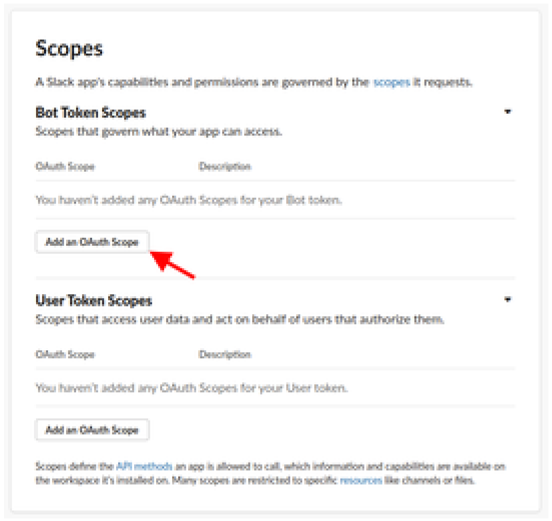
Redirect URLs

You haven't added any Redirect URLs

In the **Bot Token Scopes** section, click on the **Add an OAuth Scope** button, then add the following permissions:

- **users.profile:read**: This allows reading users profile in the workspace
- **chat:write**: This authorizes the Bot to send messages

This creates a new **Bot** token type when installing the application later.



Scopes

A Slack app's capabilities and permissions are governed by the scopes it requests.

Bot Token Scopes

Scopes that govern what your app can access.

OAuth Scope	Description
You haven't added any OAuth Scopes for your Bot token.	

User Token Scopes

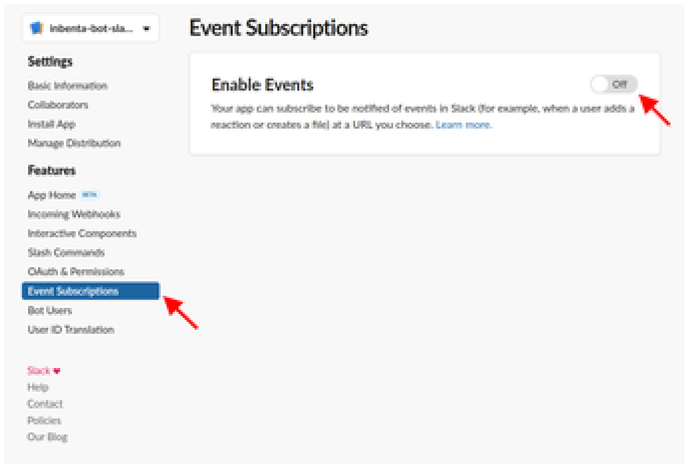
Scopes that access user data and act on behalf of users that authorize them.

OAuth Scope	Description
You haven't added any OAuth Scopes for your User token.	

Scopes define the API methods an app is allowed to call, which information and capabilities are available on the workspace it's installed on. Many scopes are restricted to specific resources like channels or files.

Event subscriptions

Click on the **Event Subscriptions** on the left menu, then activate it.



Event Subscriptions

Enable Events

Your app can subscribe to be notified of events in Slack (for example, when a user adds a reaction or creates a file) at a URL you choose. Learn more.

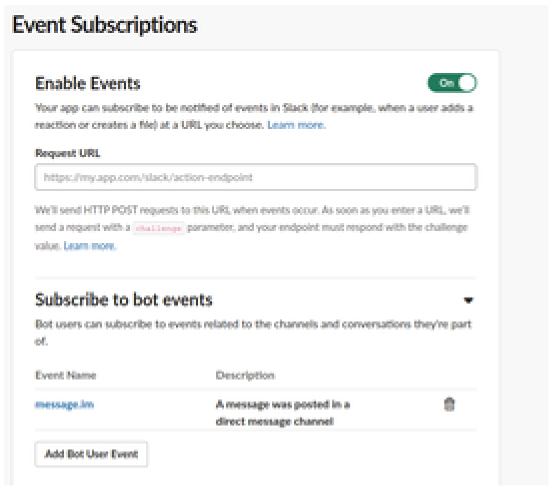
In the **Subscribe to bot events** sub-section, add the following event:

- **message.im**: This is triggered when a user sends a private message to the Bot

Enter the URL of the Bot in the **Request URL** input (URL where the app is installed).

When clicking outside of the input, a green message “**Verified**” should appear at the right of the input.

Click on the “**Save Changes**“ at the bottom on the screen.



Event Subscriptions

Enable Events

Your app can subscribe to be notified of events in Slack (for example, when a user adds a reaction or creates a file) at a URL you choose. Learn more.

Request URL

https://my.app.com/slack/action-endpoint

We'll send HTTP POST requests to this URL when events occur. As soon as you enter a URL, we'll send a request with a `challenge` parameter, and your endpoint must respond with the challenge value. Learn more.

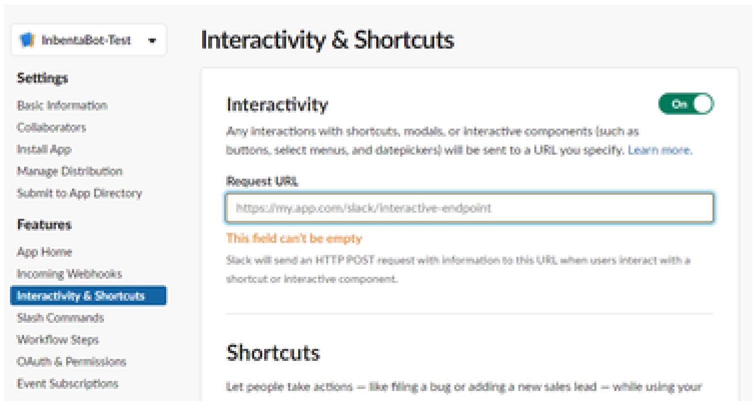
Subscribe to bot events

Bot users can subscribe to events related to the channels and conversations they're part of.

Event Name	Description
message.im	A message was posted in a direct message channel

Interactive Components

Go to the **Interactive & Shortcuts** on the left menu and activate the option, then enter the same URL as before (URL where the app is installed).



Interactivity & Shortcuts

Interactivity

Any interactions with shortcuts, modals, or interactive components (such as buttons, select menus, and datepickers) will be sent to a URL you specify. Learn more.

Request URL

https://my.app.com/slack/interactive-endpoint

This field can't be empty

Slack will send an HTTP POST request with information to this URL when users interact with a shortcut or interactive component.

Shortcuts

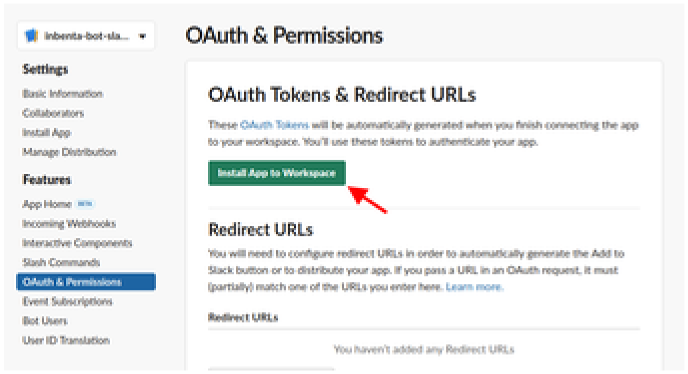
Let people take actions — like filing a bug or adding a new sales lead — while using your

Installing App on the Workspace

Return to the **OAuth & Permissions** section on the left menu, following the added **message.im** event subscription, a new Scope should've appeared:

- **im:history**: Necessary for the **message.im** event subscription, this allows reading direct messages sent to the Bot

Click on the **Install App to Workspace** button.



OAuth & Permissions

OAuth Tokens & Redirect URLs

These OAuth Tokens will be automatically generated when you finish connecting the app to your workspace. You'll use these tokens to authenticate your app.

Redirect URLs

You will need to configure redirect URLs in order to automatically generate the Add to Slack button or to distribute your app. If you pass a URL in an OAuth request, it must (partially) match one of the URLs you enter here. Learn more.

Redirect URLs

You haven't added any Redirect URLs

Get the **Bot User OAuth Token** and copy and paste in **conf/custom/slack.php** file:

```
return [
    'access_token' => ''
];
```