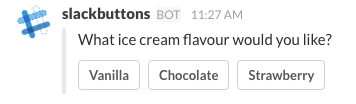
**Interactive Messages on Slack**

Simplify complex workflows and empower users to take decisive action by adding interactive buttons to your messages in Slack. This guide will demonstrate how you can create these Interactive Messages in Slack using Gupshup. To enable Interactive Messages on your app, you will need to configure your Action URL. See Note below on how to do so.

This is what an interactive message on Slack looks like:



For reference read the [Slack API page](https://api.slack.com/docs/message-buttons) for a comprehensive document on Interactive Messages.

Each message can contain a maximum of 5 buttons

Here's the JSON for an Interactive Messages - A Survey:

var question = {

"type":"survey",

"question":"What ice cream flavour would you like?",

"options":["Vanilla","Chocolate","Strawberry"],

};

When the user clicks on any option, the option is sent to the bot in the form of a message. Thus the response to a click can also be handled in the *MessageHandler()*function.

Here's an example of how your *MessageHandler()* function would look like:

function MessageHandler(context, event) {

if(event.message == "slack") {

var question = {

"type":"survey",

"question":"What ice cream flavour would you like?",

"options":["Vanilla","Chocolate","Strawberry"],

};

context.sendResponse(JSON.stringify(question));

return;

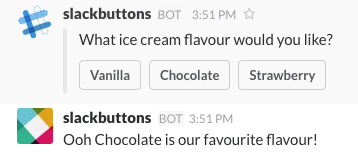
}

else if (event.message == "Chocolate") {

context.sendResponse("Ooh Chocolate is our favourite flavour!");

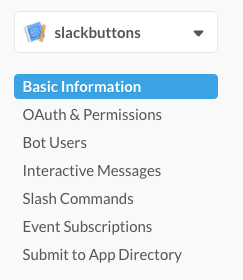
}

}

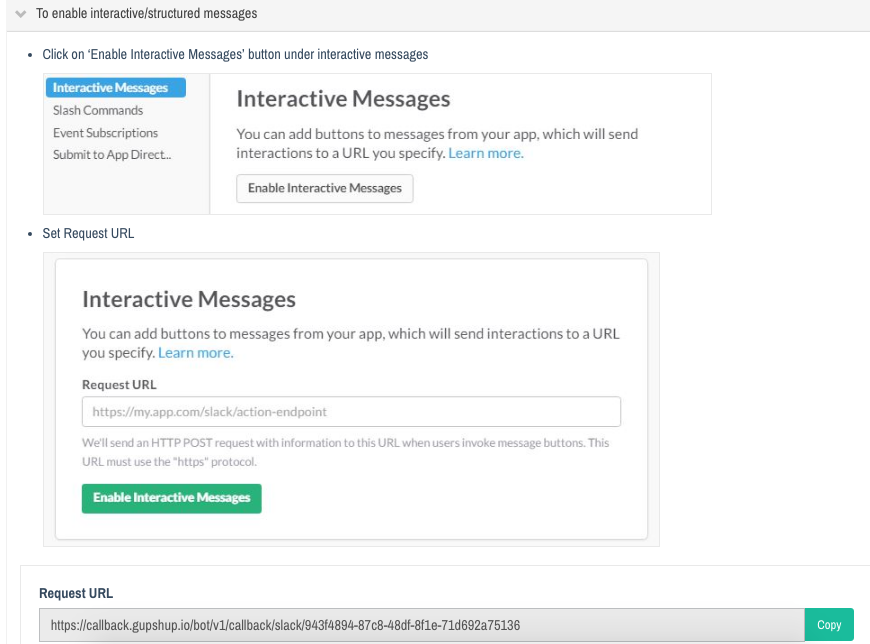


**Note:**

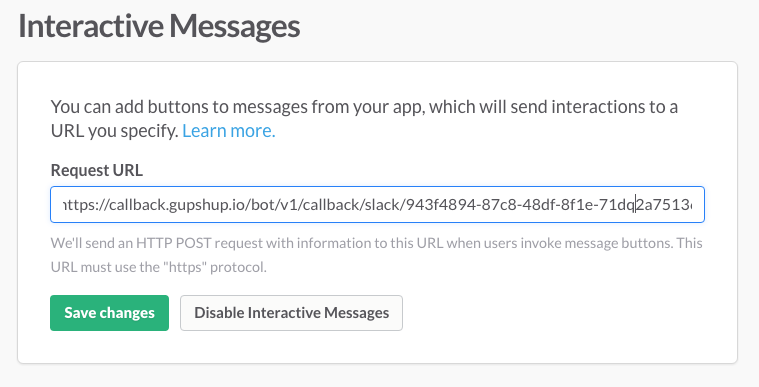
To enable Interactive Messages on your app, you will need to configure your Action URL. To do this, navigate to your application management tool and click on the 'Interactive Messages' section in the left menu.



Here you'll find a interface for setting your Action URL. To get this URL, go the Publish section of the Gupshup Bot Builder. Choose Slack Publish. Scroll down to the 'To enable interactive/structured messages' section and copy the Request URL.



Paste this URL in Slack and hit 'Save Changes'. Your application is now configured for Interactive Messages.



**Carousel:**

In the earlier example we demonstrated interactive message - Survey. Now lets see how to create a carousel

Here's the sample JSON for an Interactive Messages - carousel:

{

"type": "catalogue",

"msgid": "current\_cat123",

"items": [

{

"title": "Cosmo Royale Current account Scheme",

"subtitle": "Current Account Scheme",

"imgurl": "http://www.the-wau.com/timthumb.php?src=http://www.the-wau.com/uploads/1488790141189.png&h=523&w=1000&zc=1",

"options": [

{

"type": "text",

"title": "Know More"

}]

},

{

"title": "Cosmo Premium Plus Current account Scheme",

"subtitle": "Current Account Scheme",

"imgurl": "http://taitcc.com/sites/default/files/styles/super\_large/public/portfolio/job\_alert.jpg?itok=GznJBhMa",

"options": [

{

"type": "text",

"title": "Know More"

}

]

},

{

"title": "Cosmo Premium Current account Scheme",

"subtitle": "Current Account Scheme",

"imgurl": "https://static1.squarespace.com/static/55afaa72e4b034faade44cf7/t/58d184ab2e69cfc57137ead0/1490126040170/",

"options": [

{

"type": "text",

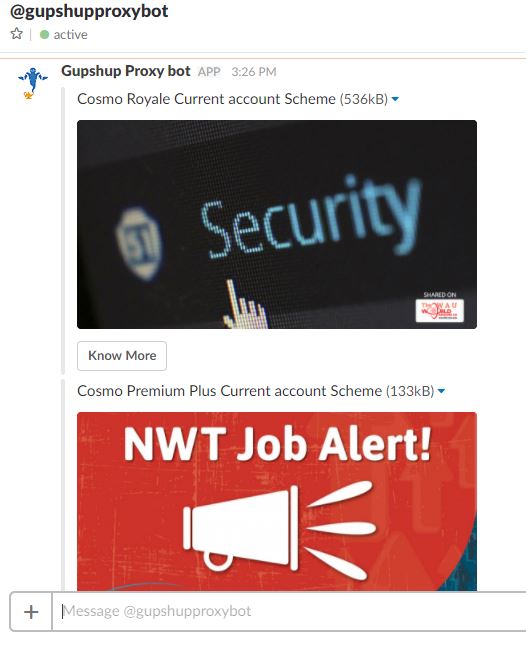
"title": "Know More"

}]

}

]

};



# How to build a bot and publish it to Slack

This guide will illustrate how you can use Gupshup to build, deploy, test and publish a bot to Slack.

## 1. Build

If you aren't familiar with Gupshup's IDE Bot Builder tool, do read our [earlier guides](https://www.gupshup.io/developer/docs/bot-platform/guide/a-hello-world-bot) to get acquainted. For this demo, let us create a simple chatbot that asks the user for his/her favourite publication and then prints the day's top stories from that publication. We will then publish this bot to Slack.

Lets start by creating a new bot on the IDE Bot Builder and calling it TechNews. Begin by asking the user for their publication preference. We can do this in the MessageHandler() method that [handles all conversations](https://www.gupshup.io/developer/docs/bot-platform/guide/handling-conversations) with the bot.

if(event.message.toLowerCase() == "hi"){

context.sendResponse("Hey there " + event.senderobj.display + " Do you prefer reading Wired or TechCrunch?");

}

We can set the user's preference based on the reply

else if((event.message.toLowerCase() == "wired") || (event.message.toLowerCase() == "techcrunch")){

setPreference(event.message);

}

We're going to use [data persistence](https://www.gupshup.io/developer/docs/bot-platform/guide/adding-basic-data-persistence) to store the user's preference. There are two types of data persistence: Botleveldata and Roomleveldata. The former stores common data for the bot for all its users across all channels while the latter stores data for each user on each messaging channel. Here since it's a user preference unique to Slack, we will use roomleveldata.

function setPreference (pref){

context.simpledb.roomleveldata.publication = pref;

context.sendResponse("Type 'news' to get latest headlines on " + context.simpledb.roomleveldata.publication);

}

Once the user's preference is set, we can make a HTTP call to the publication's RSS feed. The RSS feed returns 10 top stories from the publication. The Gupshup Bot Builder facilitates [making HTTP calls](https://www.gupshup.io/developer/docs/bot-platform/guide/making-http-calls) and handling their responses. Here we will use the .makeGet property to make a GET call whenever a user types in the word 'news'. In your MessageHandler() method type in:

else if (event.message.toLowerCase() == "news" ) {

//makes a GET call to https://api.rss2json.com/v1/api.json?rss\_url=https://wired.com/feed

context.simplehttp.makeGet('https://api.rss2json.com/v1/api.json?rss\_url=https%3A%2F%2F'

+ context.simpledb.roomleveldata.publication + '.com%2Ffeed');

}

The HttpResponseHandler() method is the method that handles responses from any HTTP calls made using the Bot Builder. The above URL returns a JSON object which needs to be parsed. This JSON object contains 10 articles each with attributes such as 'title', 'author', 'link' and more. For this demo we will display the title and the web link for a randomly chosen article.

function HttpResponseHandler(context, event) {

var respJson = JSON.parse(event.getresp); //parses the response

var stories = respJson.items;

var resp = "";

// Chose a random article from the parsed response

var randomnumber = Math.floor(Math.random() \* (stories.length - 1 + 1)) + 1;

resp = resp + stories [randomnumber].title + "\n" + stories[randomnumber].link + "\n";

resp = resp.replace("&nbsp", "");

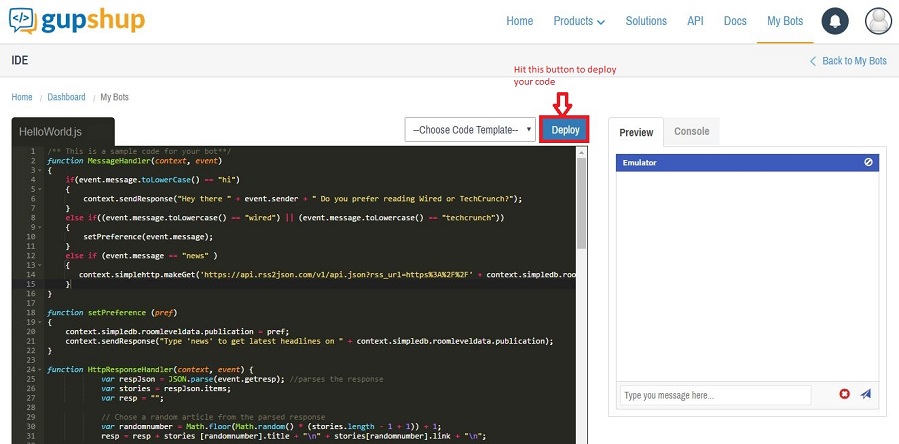
context.sendResponse(resp);

}

That's it. You have created a simple bot that will display a random news article from your favoured publication. Let us now deploy this bot.

## 2. Deploy

To deploy your bot code, hit the 'Deploy' button given on the Bot Builder tool. Once done, you will receive a confirmation for your deployment and now you are ready to test your bot using Gupshup Proxy bot.



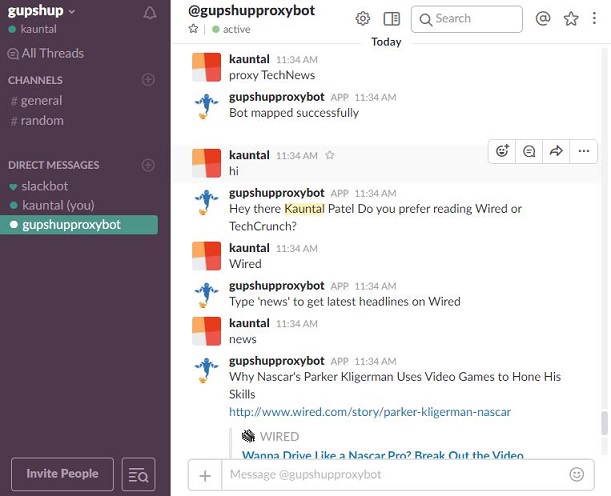
## 3. Test

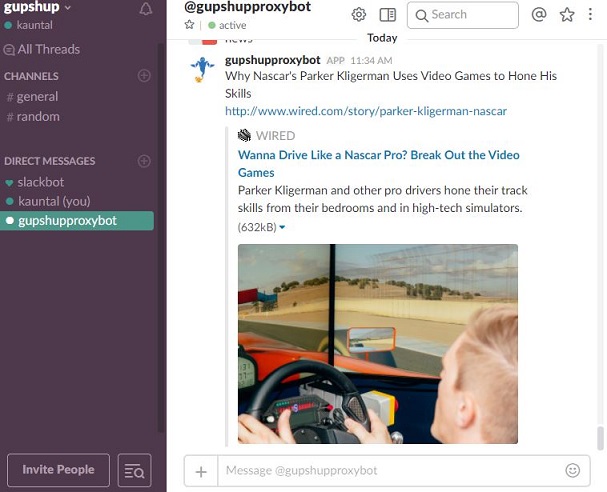
Lets test out the bot that we've built by using the [Gupshup Proxy Bot](https://www.gupshup.io/developer/demobots). The Proxy Bot is a testing tool developed by Gupshup that can mimic any bot developed on the Gupshup Bot Builder. You will find the Proxy Bot on all messaging channels such as Slack, Facebook Messenger, Telegram etc.

To access the Proxy Bot, you will have to add 'Gupshup proxy bot' to your org. Click on the 'Add to Slack' button given below:

[Add to Slack](https://slack.com/oauth/authorize?scope=bot&client_id=21071044435.25749987811&redirect_uri=http://smapi.gupshup.io/sm/api/slack/oauth/demobot1)

You can invoke a bot that you have created by using the keyword 'proxy' followed by your 'bot name'. In this case 'proxy TechNews'.





Say hi to the bot and test it out. Your chatbot is now ready to be published to Slack.