On This Page

Setting Up Your Webhook

Your webhook is the core of your Messenger bot experience. This is where your code lives, and where you will receive, process, and send messages.

In this guide, you will learn how to set up a basic webhook that supports the Messenger Platform's required webhook verification step, and is able to accept webhook events.

For more information about webhooks requirements and events, see Webhook.

Requirements

To follow this guide to set up your webhook, all you need is a computer with Node.js installed.

To deploy a live webhook that can receive webhook events from the Messenger Platform, your code must be hosted on a public HTTP server that meets has the following:

- HTTPS support
- A valid SSL certificate
- An open port that accepts `GET` and `POST` requests

Setup Steps

The Messenger Platform is Language Agnostic
To follow this guide, you will need to install Node.js, but you can write your webhook in whatever server-side language you like best..

Before you begin, make sure your server meets all of the requirements listed above.

1 Create a new Node.js project

Run the following on the command line to create the needed files and dependencies:

```
mkdir messenger-webhook         // Creates a project directory
cd messenger-webhook            // Navigates to the new directory
touch index.js                  // Creates empty index.js file.
npm init                        // Creates package.json. Accept default for all qu
npm install express body-parser --save // Installs the express.js http server fram
                                               // and then adds them to the depend
```

If everything went well, your `messenger-webhook` directory should look like this:

```
index.js
node_modules
package.json
```

2 Create an HTTP server

Add the following code to `index.js`:

```
'use strict';

// Imports dependencies and set up http server
const
  express = require('express'),
  bodyParser = require('body-parser'),
  app = express().use(bodyParser.json()); // creates express http server

// Sets server port and logs message on success
app.listen(process.env.PORT || 1337, () => console.log('webhook is listening'));
```

This code creates an HTTP server that listens for requests on the default port, or port 1337 if there is no default. For this guide we are using Express, a popular, lightweight HTTP framework, but you can use any framework you love to build your webhook.

3 Add your webhook endpoint

Add the following code to `index.js`:

```
// Creates the endpoint for our webhook
app.post('/webhook', (req, res) => {

  let body = req.body;

  // Checks this is an event from a page subscription
  if (body.object === 'page') {

    // Iterates over each entry - there may be multiple if batched
    body.entry.forEach(function(entry) {

      // Gets the message. entry.messaging is an array, but
      // will only ever contain one message, so we get index 0
      let webhook_event = entry.messaging[0];
      console.log(webhook_event);
    });

    // Returns a '200 OK' response to all requests
    res.status(200).send('EVENT_RECEIVED');
  } else {
    // Returns a '404 Not Found' if event is not from a page subscription
    res.sendStatus(404);
  }

});
```

This code creates a `/webhook` endpoint that accepts `POST` requests, checks the request is a webhook event, then parses the message. This endpoint is where the Messenger Platform will send all webhook events.

Note that the endpoint returns a `200OK` response, which tells the Messenger Platform the event has been received and does not need to be resent. Normally, you will not send this response until you have completed processing the event.

4 Add webhook verification

Add the following code to `index.js`:

```
// Adds support for GET requests to our webhook
app.get('/webhook', (req, res) => {

  // Your verify token. Should be a random string.
  let VERIFY_TOKEN = "<YOUR_VERIFY_TOKEN>"

  // Parse the query params
  let mode = req.query['hub.mode'];
  let token = req.query['hub.verify_token'];
  let challenge = req.query['hub.challenge'];

  // Checks if a token and mode is in the query string of the request
  if (mode && token) {

    // Checks the mode and token sent is correct
    if (mode === 'subscribe' && token === VERIFY_TOKEN) {

      // Responds with the challenge token from the request
      console.log('WEBHOOK_VERIFIED');
      res.status(200).send(challenge);

    } else {
      // Responds with '403 Forbidden' if verify tokens do not match
      res.sendStatus(403);
    }
  }
});
```

This code adds support for the Messenger Platform's webhook verification to your webhook. This is required to ensure your webhook is authentic and working.

The verification process looks like this:

1. You create a verify token. This is a random string of your choosing, hardcoded into your webhook.
2. You provide your verify token to the Messenger Platform when you subscribe your webhook to receive webhook events for an app.
3. The Messenger Platform sends a `GET` request to your webhook with the token in the `hub.verify` parameter of the query string.
4. You verify the token sent matches your verify token, and respond with `hub.challenge` parameter from the request.
5. The Messenger Platform subscribes your webhook to the app.

5 Test your webhook

Now that you have all the code in place for a basic webhook, it is time to test it by sending a couple sample requests to your webhook running on localhost.

1. Run the following on the command line to start your webhook on localhost:

```
node index.js
```

2. From a separate command line prompt, test your webhook verification by substituting your verify token into this cURL request:

```
ıub.verify_token=<YOUR_VERIFY_TOKEN>&hub.challenge=CHALLENGE_ACCEPTED&hub.mode=
```

If your webhook verification is working as expected, you should see the following:

- `WEBHOOK_VERIFIED` logged to the command line where your node process is running.
- `CHALLENGE_ACCEPTED` logged to the command line where you sent the cURL request.

3. Test your webhook by sending this cURL request:

```
vebhook" -d '{"object": "page", "entry": [{"messaging": [{"message": "TEST_MESS
```

If your webhook is working as expected, you should see the following:

- `TEST_MESSAGE` logged to the command line where your node process is running.
- `EVENT_RECEIVED` logged to the command line where you sent the cURL request.

6 Deploy your webhook

SSL Required
Up until now you have tested your webhook over HTTP. Your webhook must be deployed to a server with a valid SSL certificate, so that it can accept requests over HTTPS.

Your webhook should now be tested and working as expected, but it does you no good running on localhost! The next step is to deploy your webhook code to the server of your choice. This can be a cloud-based service like a Heroku or AWS EC2 instance, or your own hardware.

Once your webhook is deployed, try running the test cURL requests above again to make sure everything is working as expected. Do not forget to substitute the public URL or IP address of your server for localhost in the sample requests.

7 Subscribe your webhook to a Facebook app

Your webhook is now ready to receive event from the Messenger Platform! One problem, it is not subscribed to receive webhook events for any apps. We will cover that next in our app set up guide!

Set Up Your App

Like 115 Share

Messenger Platform