# How to make a simple Callback using Node.js

This example will not cover the creation of Apifonica account and getting a server with Node.js. We assume that you already have it.

## Step 1: Setting up packages

You will need NPM and following packages:

* express
* ejs
* body-parser
* request
* js2xmlparser

To install them, simply run in console:

*npm install express ejs body-parser request js2xmlparser*

Or download this file (package.json) and run

*npm install*

## Step 2: Setting up folder structure

After installing the packages, create the folder named “views”, with 3 files inside: header.ejs, footer.ejs and index.ejs.

---------------------------------------------------------------------------

**header.ejs**

<!DOCTYPE html>

<html>

<head>

<title>Apifonica Callback Example</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">

</head>

<body>

---------------------------------------------------------------------------

**footer.ejs**

</body>

</html>

---------------------------------------------------------------------------

**index.ejs**

<% include header %>

<div class="container">

<h1>Apifonica Callback Example</h1>

<form class="form-inline" action="/callback" method="POST">

<input type="text" class="form-control" name="client\_number" placeholder="Phone number">

<button type="submit" class="btn btn-success">Make Call</button>

</form>

</div>

<% include footer %>

---------------------------------------------------------------------------

As you can see, we use Bootstrap to style our form.

## Step 3: Declaring the variables

Make a new file in your callback directory:

*app.js*

Add the following code:

var express = require("express");

var bodyParser = require("body-parser");

var js2xmlparser = require("js2xmlparser");

var request = require('request');

var app = express();

app.set("view engine", "ejs");

app.use(express.static("public"));

app.use(bodyParser.urlencoded({extended: true}));

app.use(bodyParser.json());

//SERVER

app.listen(process.env.PORT, process.env.IP, function(){

console.log("Server is listening!!!");

});

Now try to run your server: type “node app.js” in the console.

You should see the message “Server is listening!!!” in the console.

## Step 4: Create an application and rent a number.

Go to your account at Apifonica web site, rent a local number of your choice here: <https://account.apifonica.com/numbers/buy/>

Your clients will see this number as a Caller ID.

Also, you need to create a new Application here: <https://account.apifonica.com/applications/new/>

While creating the application, type in your server’s address and “***/callback/getxml***” at the end. Leave the POST method.

Please save the application SID and the number.

After you do all this, add the following code to app.js:

//YOUR AUTHENTICATION DATA

const accountSid = "HERE GOES YOU ACCOUNT SID";

const authToken = "HERE GOES YOU AUTH TOKEN";

const callbackAppSid = "HERE GOES AN APPLICATION SID";

//YOUR CONTACT PHONE NUMBER (any number which will be called first) AND CALLERID (must be rented from Apifonica)

const myNumber = "HERE GOES YOUR REAL PHONE NUMBER";

const callerID = "HERE GOES VIRTUAL NUMBER WHICH WILL APPEAR AS CALLER ID";

Fill in the values and you are ready to go. AccountSID and AuthToken you will find in your personal account.

## Step 5: Set up routers

Add the following code to app.js:

//OBJECT FOR XML CONSTRUCTION (we will convert it to XML and feed to Apifonica. You can change it for different call behaviour.)

var cbObj =

{

makeCall: {

number: ""

}

};

//ROUTERS

//THIS ROUTER SIMPLY RENDERS THE PAGE WITH A FORM

app.get("/callback", function(req, res) {

console.log("Someone is requesting your callback page. Rendering index.html...");

res.render("index");

});

//THIS ROUTER GETS FORM DATA AND MAKES A REQUEST TO APIFONICA TO INITIATE A PHONE CALL

app.post("/callback", function(req, res) {

console.log("Client entered following phone number: " + req.body.client\_number + " and asked for a callback.");

//SAVING CLIENT'S NUMBER

cbObj.makeCall.number = req.body.client\_number;

//MAKING A REQUEST TO APIFONICA

request.post({

url: "https://" + accountSid + ":" + authToken + "@api.apifonica.com/v2/accounts/" + accountSid +"/calls",

headers: {'content-type': 'application/json'},

json: {

'from': callerID,

'to': myNumber,

'call\_app\_sid': callbackAppSid,

'events': "all"

}

},

function(error, response, body) {

if(!error) {

console.log("APIFONICA RESPONSE:");

console.log(response.body);

} else {

console.log("ERROR:");

console.log(error);

}

});

res.redirect("/callback");

});

//THIS ROUTER HANDLES THE CALL FLOW AND GETS UPDATES ON CALL EVENTS

app.all("/callback/getxml", function(req, res){

if(req.body.status.toString() != 'queued') {

console.log("APIFONICA EVENT UPDATE:");

console.log(req.body);

//PLEASE NOTE THAT IF WE DO NOT WANT TO CHANGE THE CALL FLOW, WE ANSWER WITH AN EMPTY STRING AS FOLLOWS

res.send("");

} else {

//SO IF REQUEST STATUS IS "QUEUED" AND APIFONICA WAITS FOR INSTRUCTIONS, WE CONSTRUCT AN XML FILE FROM THE OBJECT ABOVE

res.send(js2xmlparser.parse("response", cbObj));

console.log("APIFONICA REQUEST:");

console.log(req.body);

console.log("SERVER RESPONSE:");

console.log(js2xmlparser.parse("response", cbObj));

}

});

## Step 6: Give it a shot!

Save everything and run the server. Your simple callback should work.