Extensions Developer Rig

Introduction

The Developer Rig allows Extensions developers to develop and test Extensions quickly, easily, and locally. It is a lightweight web app that runs in a browser.

The developer rig can be used in two modes:

- *Local mode* lets you get started building quickly, without going through Extensions developer onboarding. Local mode gives you access to mock versions of the APIs and Twitch PubSub.

- *Online mode* lets you test with production APIs and hosted assets on Twitch. To use online mode, you must first complete Extensions developer onboarding (see Onboarding in *Getting Started with Extensions*). In online mode, valid client IDs and extension secrets are required, and these can be generated only after onboarding is complete.

The Developer Rig also can be used to configure your extension to be Bits enabled.

The Developer Rig is supported in Chrome and Firefox on macOS and Windows. Make sure your version of the rig is up to date.

© 2018 Twitch Interactive, Inc. Confidential and proprietary.

Starting Up the Rig

To get the Developer Rig:

1. Download and extract the Zip file
2. Open the `scripts` folder in the `developer-rig` folder.
3. Run the script: double-click the `configure` (macOS) or `configure.cmd` (Windows) file. This script installs and configures all dependencies for the rig.

After installing the rig, you can launch it by either:

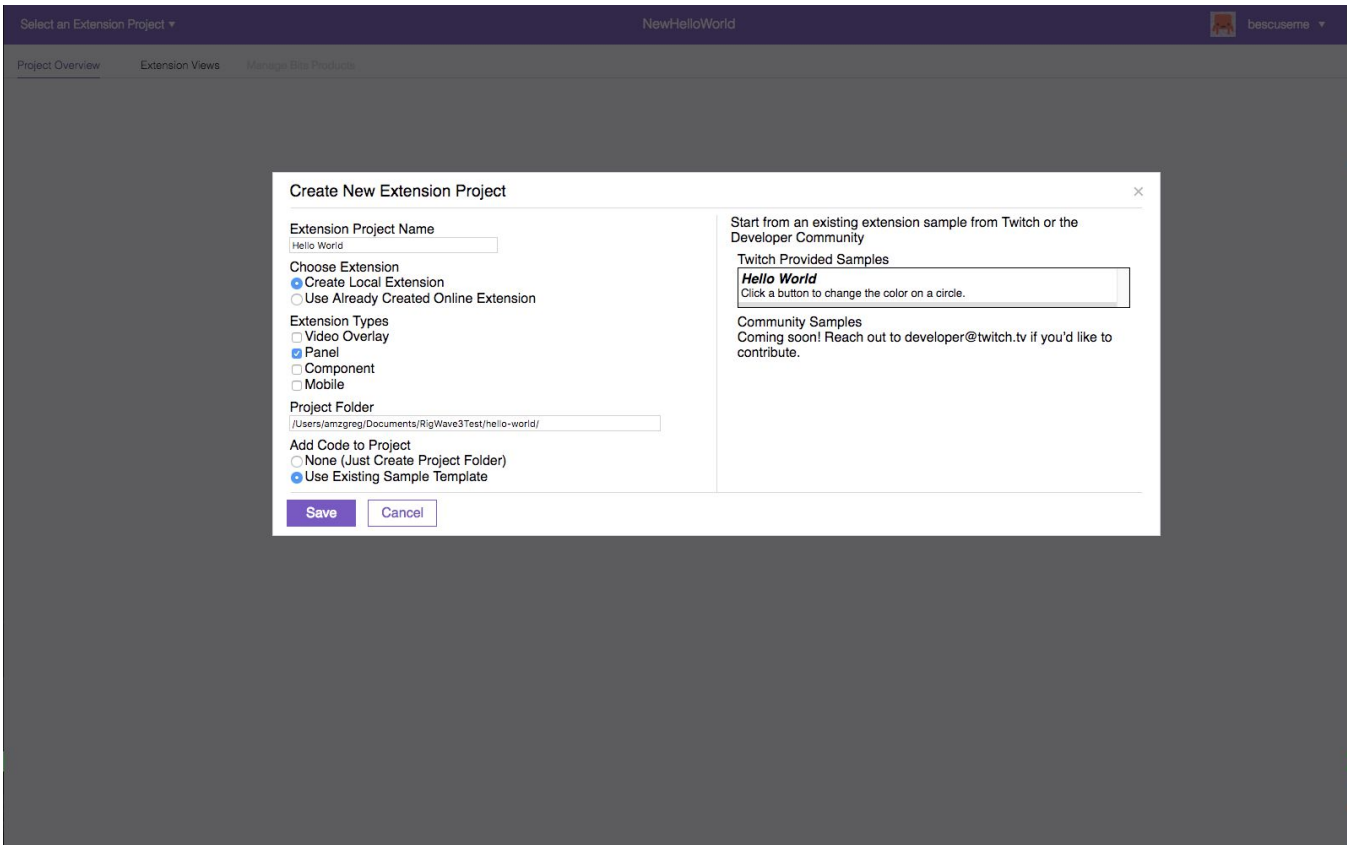
- Navigating to the root of the rig's folder and typing `yarn start` in your command line.
- Executing the run script in the rig's scripts folder.

Then sign in with your Twitch credentials.

Creating a Project

From the rig, Select **Select an Extension Project > Create New Project**.

At the center of the Developer Rig is your extension *project*, a combination of the extension code and the extension manifest (metadata needed to run the code on Twitch or in the rig). When you start the rig for the first time, you'll need to create your first project.



To create an extension project:

1. Specify the **Extension Project Name**.

2. Under **Choose Extension**, check one of the following:
    - **Create Local Extension**, to use an extension manifest created locally with the rig. Using a local extension manifest is ideal for getting started and experimenting with Extensions.
    - **Use Already Created Online Extension**, to use an extension manifest already created on your Twitch developer dashboard. Using an extension manifest created on the Twitch dev site enables you to run against production APIs.

3. If you chose **Use Already Created Online Extension**, enter the **clientId** (copy this from your Extensions overview page), **secret** (copy this from **Settings > Secret Keys** for your extension), and extension **version number** (the default is 0.0.1).

4. If you chose **Create Local Extension**, then under **Extension Types**, select one or more types (video overlay, panel, video component, or mobile). (Ignore this if you create your extension manifest on the Twitch dev site, as the extension type is pulled in automatically.)

5. Specify the root **Project Folder** for project files.

6. Under **Add Code to Project**, indicate how you want to add the initial code: **None** (the rig create just a project folder, no code) or **Use Existing Sample Template** (full code samples provided by Twitch, with front-end and, in some cases, back-end parts). If you choose to use an existing template, select a template.

7. Click **Save** or **Cancel**.

Other Rig Functionality

The rig has several tabs to assist your development process and run your project:

Tab	This is where you ...
Project Overview	Run your project (following the steps on the page), manage project metadata, and run the commands to host your front- and back-end files. Also, click here to refresh your extension project, if you change your extension manifest (and it was created on the Twitch dev site, not locally).
Extension Views	Create and interact with different views of your extension, for testing. Select among these **Extension Types**: ● The **Panel** view shows what your extension looks like as a panel extension.. ● The **Configuration** view appears on a broadcaster's dashboard for an extension. ● The **Dashboard** view is presented to broadcasters after they install an extension but before they activate it. To add a new view for the configured client ID, click the **+** button. To delete a view, click the **X** on the view. Depending on the selected type, you also select one or more **Viewer Types**: ● Broadcaster ● Logged-in viewers (linked and unlinked) ● Logged-out viewers (i.e., anonymous users) Extension output logs can be redirected to the rig console, a local console specific to the developer rig. The console is at the bottom of the Extension Views page. To enable your extension to output to this console, make the following call from the extension's JavaScript front end: `window.Twitch.ext.rig.log(<message to log>)`
Manage Bits Product	Integrate Bits into your extension. Before this tab is available for use, you must: ● Complete Extensions developer onboarding (see Onboarding in *Getting Started with Extensions*). ● Include Bits monetization in your extension (see Bits in Extensions Guide & Reference).

Using Local Mode

Local mode enables you to:

- Run your extension against mock APIs and mock PubSub locally (on your machine). In this way, you can start building Extensions without first doing Twitch Extensions developer onboarding.

- Perform integration tests against your extension via configurable responses to the Extensions helper library.

Using the Run List to Trigger Mock Responses

The foundation of local mode is a JSON document called the “run list.” It gives developers the ability to trigger specific callback responses through the Extensions helper library. To your extension, it appears like these responses came from Twitch Extensions APIs.

The local-mode edition of the rig has a run list pre-populated with mock responses. You can alter these responses by editing the document.

When local mode is enabled, each extension view has a dropdown that enables the developer to choose a response from the run list. When a response is selected, clicking **Trigger** sends the response through the extension coordinator, which passes it through the helper to your extension code.

Different views can point to different places in the run list at the same time.

To edit the run list, look for `runlist.json` in your rig directory. Each entry has three components:

- A name for easy identification
- The type of callback (e.g., `onContext`)
- The callback response

Before editing and creating new responses, see the Extensions JavaScript helper documentation.

Providing a Secret for Your EBS in Local Mode

When you use local mode with your EBS, you'll be using a hard-coded secret in the rig. This is included in the Hello World example. When creating an extension on Twitch, you'll have your own client ID and secret.

The base-64-encoded local mode secret is:
`kkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk`

The local-mode secret is provided to speed up initial development. While using it, do not expose your EBS beyond your local machine.

Using Mock PubSub

When you are in local mode, to make calls to PubSub, it's important to make sure your EBS is pointing at the correct URI. To correctly send your message, make calls to:

```
localhost.rig.twitch.tv:3000/extensions/message
```

not to:

```
api.twitch.tv/extensions/message
```

Format the rest of your message the same as if you were sending to Twitch PubSub. For examples, see the Hello World `backend.js` files.

TROUBLESHOOTING

I'm sure my Developer Rig configuration is correct, but my extension doesn't work

Clear your browser cache and local storage and restart the developer rig.

To delete the rig's local storage open the javascript console in your browser on a tab with the rig open and do `localStorage.clear();` then refresh.

Ensure you've included the Twitch Extensions Helper in your front-end files.

I get an error when trying to run yarn test

Sometimes deleting and reinstalling your `node_modules` fixes this issue.

On macOS, you may need to explicitly install watchman via `brew install watchman`.

yarn install fails in libssh2

See issue #48. Be sure `libssh` and its dependencies are installed.

I get an error when pulling in the example project

Ensure that Git is in your PATH variables by trying to run `git` at your command line. If that works, also ensure that the local folder does *not* currently exist.

The Developer Rig stops running unexpectedly in local mode

Try deleting the `node_modules` directory and rerunning `yarn install`.

I created my extension manifest on the Twitch dev site but can't find my front-end files

If you created your extension manifest on the Twitch dev site, you must specify your **Base Testing URI** as https://localhost.rig.twitch.tv:8080. This field is discussed under Asset Hosting, in *Releasing & Maintaining an Extension*.