Assignment Report

Antonino Santa Rosa
Politecnico di Torino

October 2022

Contents

1 The Extension

1.1 Introduction

An add-on is a collection of code that the gateway runs to gain new features. There are three primary classes of add-ons: adapter, notifier, and extension. Since the assignment requires some new features to be added to the interface of the gateway, the add-on class of our interest is the **extension** one.

An *extension* add-on a has one primary component, an extension, and an optional API handler. The **extension object** is what is actually loaded by the web interface at run-time, to provide the desired new functionality. An **API handler** is a back-end piece that can extend the gateway's REST API. This could allow for even more functionality from a UI extension than only what is provided by the gateway's API.

1.2 Implementation

The implementation flow follows the example of an extension provided by Michael Stegeman and suggested in the official WebThings documentation.

The extension extends the `Extension Class`, which is global to the browser window. This allows to have access to the APIs of the gateway.

Furthermore, an add-on can register a new APIHandler so that when an authenticated request is made to `/extensions/<extension-id>/api/*`, the custom API handler will be invoked with request information and it should send back the appropriate response.

The aim of the extension basically is to permit to get, in JSON format, the list of the installed add-ons on the gateway and/or exfiltrate this list on Dropbox after providing the "authentication token" by filling the corresponding field of the form, then echo the result back as JSON.

1.2.1 Frontend

The frontend `HTML` part of the extension can be found in `/views/content.html`. It is loaded in the Extension Class from the server and it is not a full document, but rather a snippet, since it is used to fill in a `<section>` tag. This can be done synchronously within the JavaScript, but it is nicer to keep the view content separate. Then, the *manifest* for this add-on instructs the gateway which resources to load, and which are allowed to be accessed via the web.

1.2.2 APIHandler

Basically, the APIHandler handles the `POST` requests containing the Authentication Token and the `JSON` format of the data to be exfiltrated i.e., the add-ons list. Then it exploits the Dropbox APIs to perform the upload of the list in a Dropbox remote folder accessible through the provided token. The gateway-addon library provides wrappers for the API requests and responses.

1.2.3 Extension Class

The code inside `extension.js` basically:

- Adds a top-level menu entry for the extension in the side bar of the gateway.
- Loads some HTML asynchronously from the server.
- Sets up two event listeners. One on the "List" button to get and display the list of the installed add-ons in `JSON` format, the other on the "Exfiltrate" button to perform the list exfiltration after the input of a valid token and the consequent confirmation.

1.3 Test the Extension

The first time that the Gateway is started with `npm install && npm start`, it is **built** and then run. However, to test the extension, some fixes are required. Therefore, after the `npm install` it won't follow the `npm start`.

1.3.1 Setting up the Gateway

From within the project `gateway` folder:

1. `npm install`
2. `npm run build`
3. Inside `build/controllers/extensions_controller.js`, the line 64:

   ```
   const rsp = await apiHandler.handleRequest(req);
   ```

   must be substituted by the following code[1]:

   ```
   const rsp = Object.assign(
       new APIResponse(),
       await apiHandler.handleRequest(req)
   );
   ```

Then the extension can be set up.

1.3.2 Setting up the Extension

1. The extension must be cloned inside the `~/.webthings/addons` folder:

   ```
   git clone https://github.com/ninosanta/exfiltration-extension.git
   ```

2. Download the extension modules:

   ```
   cd exfiltration-extension
   npm install
   ```

3. Inside the project `gateway` folder, the gateway must be just **run** by run:

   ```
   node build/app.js
   ```

4. Lastly, let's enable the add-on by navigating to Settings → Add-ons in the UI. Click the Enable button for "WT Exfiltration Extension".

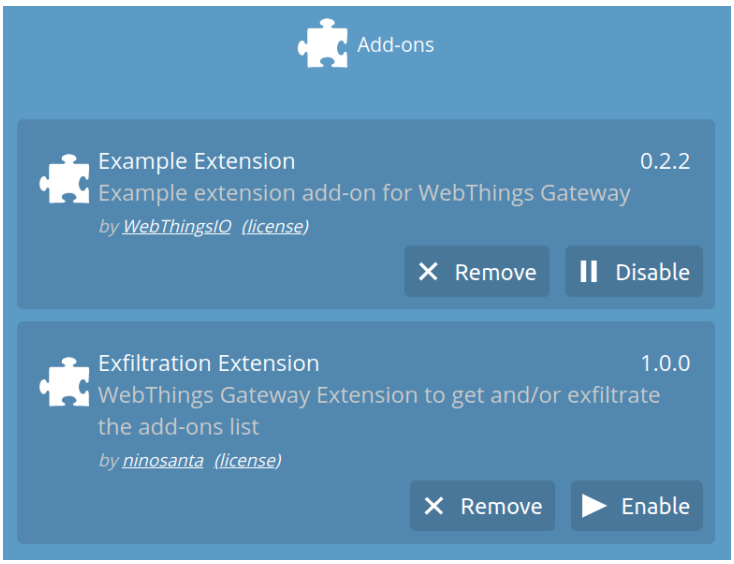
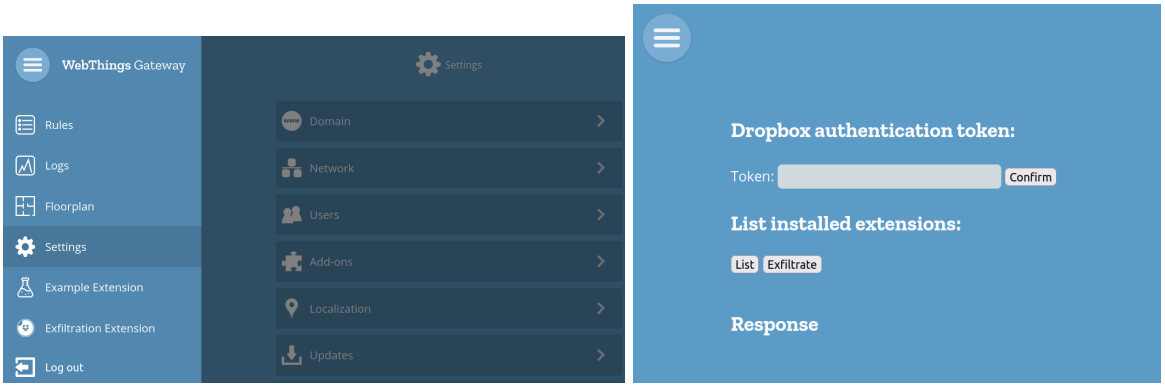


Figure 1: Activation page

5. Refresh the page to make the extension show up in the UI.



(a) Sidebar menu (b) Extension page

Figure 2: Final result

Notice that after having performed the steps listed in 1.3.1, the Gatewy must always be started by run `node build/app.js`.

2 Resources

- https://github.com/WebThingsIO/wiki/wiki/HOWTO:-Create-an-add-on
- https://hacks.mozilla.org/2019/11/ui-extensions-webthings-gateway/
- https://github.com/dropbox/dropbox-sdk-js/

References

[1] flatsiedatsie. fix rsp.getStatus() is not a function. https://github.com/createcandle/candle-controller/commit/d950cd28f26a659036658815db4137d678327d07, 2022. [Online; published 29-July-2022].