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 has one main component, the extension object, 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 APIs`. This could allow for even more functionalities from a User Interface (UI) extension with the respect to what is normally 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 will be a subclass of the `Extension Class`, which is global to the browser window. This permits 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 the request information and it should send back the appropriate response.

Basically, the aim of the extension is to permit to display, in `JSON` format, the list of all (enabled or not) installed add-ons on the gateway or exfiltrate this list on Dropbox after providing the "authentication token" by filling the corresponding field of the form.

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 a *snippet* 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 extension instructs the gateway which resources to load, and which are allowed to be accessed via the web.

1.2.2 APIHandler

Basically, the APIHandler manages the `POST` requests containing the Authentication Token and the `JSON` format of the data to be exfiltrated (i.e., the add-ons list). Then, the APIHandler 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 `/js/extension.js` basically:

- Adds a top-level menu entry for the extension in the side bar of the gateway.
- Loads the HTML frontend.
- Sets up two event listeners. One on the "List" button to get and display the list of the installed add-ons in `JSON` format by exploiting a native gateway API i.e., `getInstalledAddons()`. The other event listener is on the "Exfiltrate" button, it permits to get the add-ons list, as it happens after clicking the "List" button, and then it triggers the `APIHanlder` that will act as explained in subsubsection 1.2.2 to perform the list exfiltration on Dropbox after the input of a valid token and the consequent confirmation.

1.3 Test the Extension

To test the extension, following these steps should be enough:

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

```
cd ~/.webthings/addons
git clone https://github.com/ninosanta/exfiltration-extension.git
```

2. Download the extension modules:

```
cd exfiltration-extension
npm install  # if error on dropbox module,
             # try with a newer version of node
```

3. Inside the project `Gateway` folder:

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

4. Lastly, enable the add-on by navigating to Settings → Add-ons in the UI menu and click the Enable button for "Exfiltration Extension".

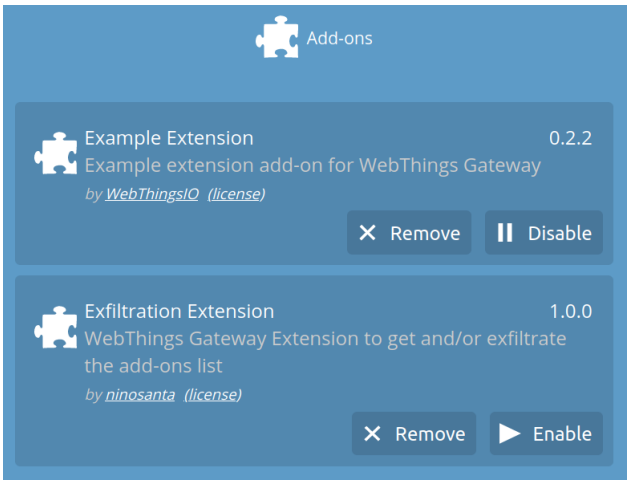
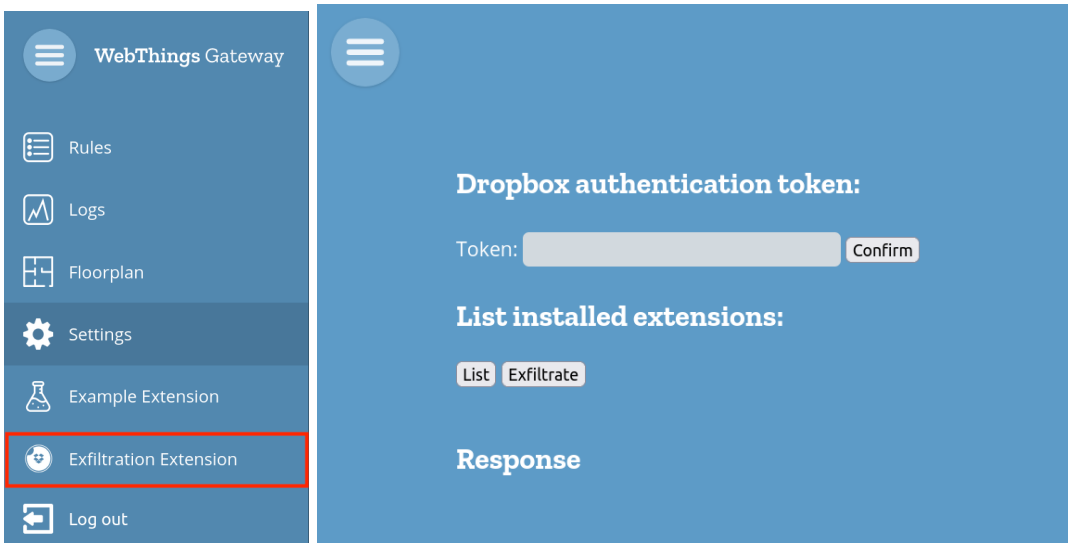


Figure 1: Activation page

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



(a) Sidebar menu entry (b) Extension page

Figure 2: Final result

Notice that for starting the Gateway with just `node build/app.js`, it is assumed that the Gateway has successfully already been started in the past.

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/