

Mozilla Addon Builder Package Building System

Piotr Zalewa

May 13, 2010, alpha 0.1

Download this document from

<http://github.com/zalun/FlightDeck/raw/master/Docs/Package%20Building%20System.pdf>

Some relevant graph slides are available

<http://github.com/zalun/FlightDeck/raw/master/Docs/Addons%20Builder%20-%20Build%20System.pdf>

1 Assumptions for the current iteration

1. Name of the Package is not unique anymore.
Packages are identified by it's *unique ID*. There may and probably often will be many Packages with the same name¹.
`/library/123456/`
2. Version is a tag.
Version is important. It is used to tag major *Revisions*. If a package is called without any Version specified (as above), the latest versioned Revision will be used.
`/library/123456/version/0.1/`
3. Revision Number is used to precisely identify a Revision.
It is completely parallel to the Package Version
`/library/123456/revision/654/`
4. No collaborative editing.
Although there will be no connection between Packages owned by different Users, design the system to not complicate future implementation of such functionality.
5. Package remembers which SDK version was used to build it.
This is very complicated also on the front-end side. It will be created during the next iteration. However, for the future implementations we will save the Jetpack SDK version number which was installed at the moment the Package was created.

¹Check if it will not make any problem with Addons and uploading to AMO

2 Logical structure

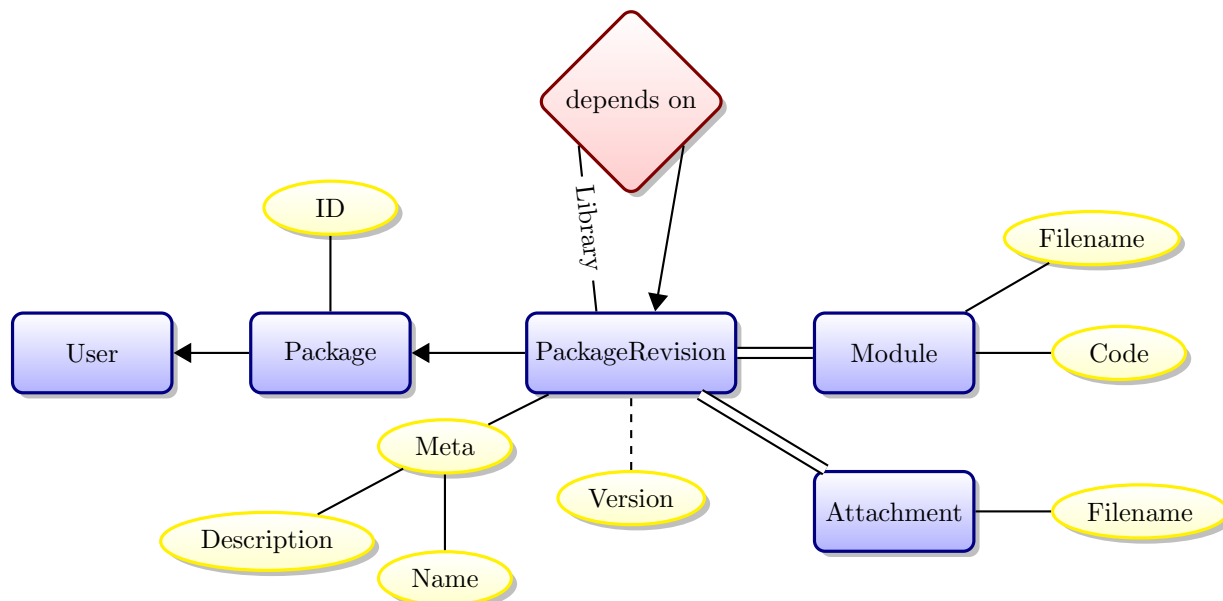


Figure 1: Database design. It does not secure unique filenames for Modules or Attachments. Additional code needs to be written to prevent duplicating filenames.

3 Export XPI

Only Addons may be used to create an XPI². Be aware that it is possible and common to export XPI from partially unsaved data. This happens when User will use the *Try in browser* functionality. In this case XPI can not be send to AMO³.

3.1 Create directory structure

Directory structure should be as close as standard Jetpack SDK as possible. Jetpack SDK should be copied to a temporary directory as more than one Addon compilation could take action at the same time. Desired revisions of Libraries and Addons will be exported into **packages** directory.

Listing 1: Parts of the tree of a copied Jetpack SDK directory.

```
1 /tmp/jetpack-sdk-{hash}/
2 |-- bin/
3 | |-- activate
4 | |-- cfx
5 | |-- [...]
6 |-- packages/
7 | |-- jetpack-core/
8 | |-- [...]
9 |-- python-lib/
10 |-- static-files/
11 |-- [...]
```

²An XPI installer module is a ZIP file that contains a Package — Manifest with all code needed to run the Addon

³<http://addons.mozilla.org/>

3.2 Export Packages with Modules

1. Create Package and its Modules directories
/tmp/jetpack-sdk-{hash}/packages/{Package:ID}/
/tmp/jetpack-sdk-{hash}/packages/{Package:ID}/lib/
2. Use collected data to create the Manifest.
/tmp/jetpack-sdk-{hash}/packages/{Package:name}/package.json
3. Create Module files
Iterate over the assigned Modules and create a ".js" file with its content inside Package's lib/
directory.
4. Export dependencies
Iterate over Libraries on which a Package depends and repeat this section (*Export the Package with Modules*) for every Library.

3.3 Build XPI

1. Set virtual environment to the temporary Jetpack SDK
2. Change directory to /tmp/jetpack-sdk-{hash}/packages/{Package:name}/
3. Call `cfx xpi`.
The {Package:name}.xpi file will be created in current directory.
4. Send location to the front-end to be used in further actions
In example calling Test in Browser.

3.4 Test in browser

To test the Addon in the Browser it is not necessary to save current code in the PackageRevision. An XPI will be build from current data displayed inside the Addon Builder⁴. Addon will be installed temporarily — it will be automatically removed after certain conditions will happen⁵. The process takes following steps:

1. Build an XPI from code in editor
2. Send a *load Addon request* to the *FlightDeck Addon*⁶
3. XPI is downloaded and installed by the Addon
4. Addon sends a remove request after the XPI is downloaded
5. Whole Jetpack SDK used to create XPI is removed

3.5 Upload to AMO

Sending the XPI's to AMO is postponed and will be coded after current iteration.

XPI needs to be created from a database object. Then **mechanize** lib will be used to login to AMO and upload the file faking it was done directly from the browser.

⁴As UI will probably be designed in the way that the Libraries will be open in different Browser Tab than the Addon, only the latter data will be taken directly from editor

⁵It is not yet decided what these conditions are. Should it happen after User will "*leave the editing environment*" or restart the browser

⁶FlightDeck Addon is a Jetpack extension allowing to temporary installation of the XPI. It needs to be called with an URL of the XPI. It is currently in *alpha* stage, there is a proposition to replace it with a more generic Addon

4 Integration with the Browser

Integration with the Browser is provided by FlightDeck Addon.

*FlightDeck
Addon is
provided and
developed by
Mozilla*

4.1 Current status

Currently there is a *proof of concept* version of that Addon. It provides the support for instant addons. These are installed temporarily until the browser will be restarted.

4.2 Try in browser

Provide a temporary instalaltion of the Addon.

Install Addon

Addon will download and install an XPI created by the Package Builder. Afterwards it will send back a command to remove the SDK. Installation should use the built-in support for instant addons.

Remove Addon

This should happen on request or automatically after a specific action will happen. It is still not decided if leaving the editing environment should be considered as a *remove addon* action.

On request means that the User has the information about temporarily installed Addon/s and is able to send request to uninstall it/them.

On browser quit — all of the Addons Iser is currently testing are gonna be removed after Browser will quit or restarted. This should happen after User will quit the Browser, terminate it or restart after crash.

On leaving the editing environment — this happens whith the onunload event fired on the Addon tab (navigating away or closing the tab)

To be continued...