

Mozilla Addon Builder

Definition of the Package Building System

Piotr Zalewa

April 30, 2010

If in doubts, please take a look at the accompanied document:

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

1 Syntax

1.1 Objects

x, y, z — *represents* $[a..z]$

m, n — *represents* $[0..9]^+$

Ux is the specific User (identified by *User:name*)

Px is the specific Package (identified by *Package:name*)

It should always be used within its **type** context as Lx — Library or Ax — Addon

Every Package has an associated PackageRevision¹ (identified by a triplet $Ux:Py.n$
User/Package/PackageRevision:revisionNumber)

Mx is the Module (identified by $Ux:Py.n:Mz$ *PackageRevision/Module:name*²)

1.2 Object identification — revision numbers and HEAD

$Ux:Py.n$ defines revision of the Package.

$Ua:La.1$ — First revision of Library La saved by Ua .

$Ux:Py.n:Mz$ defines the precise Module revision — a Module inside the PackageRevision.

$Ua:La.1:Ma$ — Module Ma inside the first revision of Library La saved by Ua .

$Px \implies Uy:Px.n$ is the HEAD revision of the Package

$La \implies Ua:La.1$ — La 's HEAD points to the first revision of Library La saved by Ua .

$Ux:Py.n \supset \{Ux:Py.m:Mz, \dots\}$ Modules inside the Package revision.

$Ua:La.2 \supset \{Ua:La.1:Ma, Ub:La.2:Mb\}$ — Second revision of Library La saved by Ua contains Ma saved by Ua in his La 's first revision and Mb saved by Ub in his second La 's revision.

¹PackageRevision is not the same as Package version. The latter is just meta-data, a text field of PackageRevision object used only in exported XPI. It will no longer be used for data identification.

²Every data object is identified by a PackageRevision. The concept is similar to *git*'s commits. In essence, for every saved Module change, a new PackageRevision is created.

2 Editing Library and its Modules

2.1 Starting point

All next scenarios start from the $Ua:La.1$ defined as follows.

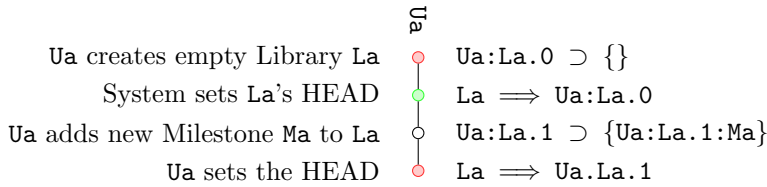
$La \Rightarrow Ua:La.1 \supset \{Ua:La.1:Ma\}$

Package La is created by User Ua .

La 's HEAD is PackageRevision identified as $Ua:La.1$

It contains only one module - Ma

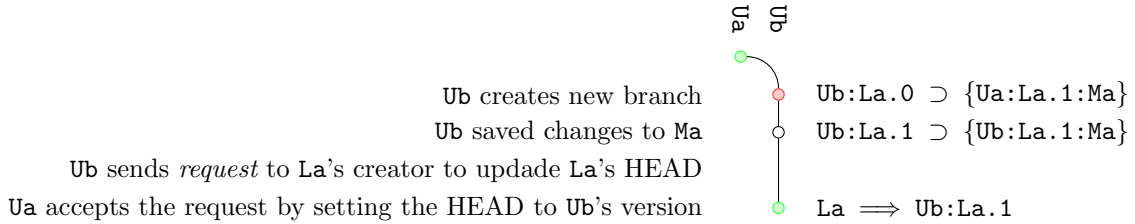
Following steps had to happen to achieve above status:



2.2 Scenario (1 Module, 2 Users, no dependencies)

Ua and Ub are working on La

Ub modified one module



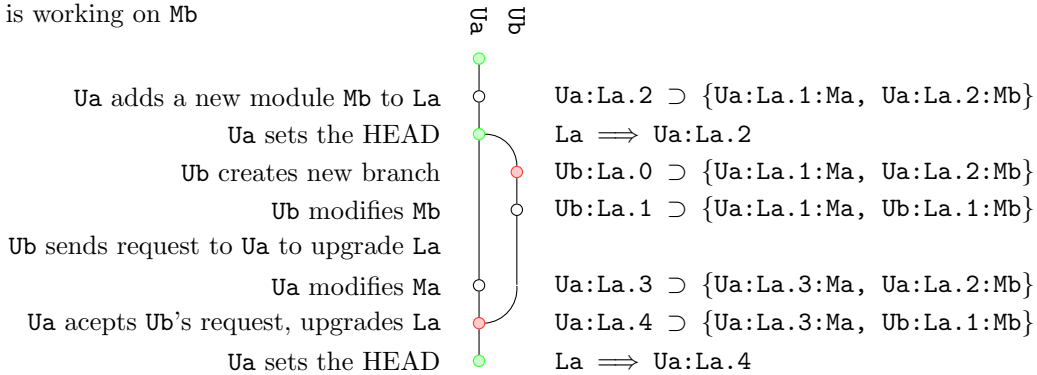
Result: $La \Rightarrow Ub:La.1 \supset \{Ub:La.1:Ma\}$

2.3 Scenario (2 Modules, 2 Users, no dependencies)

Ua and Ub are working on La

Ua created module Mb

Ub is working on Mb



Result: $La \Rightarrow Ua:La.4 \supset \{Ua:La.3:Ma, Ub:La.1:Mb\}$

2.4 Scenario (2 Modules, 2 Users, no dependencies)

Ua and Ub are working on La

Ub created module Mb

1. Ub adds a new module Mb to La
 $Ub:La.0 \supset \{Ua:La.1:Ma\}$ — automatic fork of La
 $Ub:La.1 \supset \{Ua:La.1:Ma, Ua:La.1:Mb\}$
2. Ub modifies Mb
 $Ub:La.2 \supset \{Ua:La.1:Ma, Ub:La.2:Mb\}$
3. Ub sends request to Ua to upgrade La from Ub:La.2
4. Ua modifies Ma
 $Ua:La.2 \supset \{Ua:La.2:Ma\}$
5. Ua accepts Ub's request
 $Ua:La.3 \supset \{Ua:La.2:Ma, Ub:La.2:Mb\}$
6. Ua sets the HEAD
 $La \implies Ua:La.3$
7. Result: $La \implies Ua:La.3 \supset \{Ua:La.2:Ma, Ub:La.2:Mb\}$

2.5 Scenario with conflict (2 Modules, 2 Users, no dependencies)

Ua and Ub are working on La

Ua created module Mb

Ua and Ub are working on Mb

Conflict arises...

1. Ua adds a new module Mb to La
 $Ua:La.2 \supset \{Ua:La.1:Ma, Ua:La.2:Mb\}$
2. Ua sets the HEAD
 $La \implies Ua:La.2$
3. Ub modifies Mb
 $Ub:La.0 \supset \{Ua:La.1:Ma, Ua:La.2:Mb\}$ — automatic fork of La
 $Ub:La.1 \supset \{Ua:La.1:Ma, Ub:La.1:Mb\}$
4. Ua modifies Mb
 $Ua:La.3 \supset \{Ua:La.1:Ma, Ua:La.2:Mb\}$
5. **CONFLICT**
At the time we've got two versions of La.Mb which came out from the same version
6. Ua sets the HEAD
 $La \supset Ua:La.3$
7. Ub receives info that his source is behind the HEAD
 $Ub:La.1:Mb$ (and $Ub:La.1$) is marked as *conflicted*
Ub can't send the update request
8. Ub manually solves conflict by editing the Mb and removing the *conflict flag*
 $Ub:La.2 \supset \{Ua:La.1, Ub:La.2:Mb\}$

9. U_b sends request to U_a to upgrade La from $U_b:La.2$
10. U_a accepts U_b 's request
 $U_a:La.4 \supset \{U_a:La.3:Ma, U_b:La.2:Mb\}$
11. U_a sets the HEAD
 $La \Rightarrow U_a:La.4$
12. Result: $La \Rightarrow U_a:La.4 \supset \{U_a:La.3:Ma, U_b:La.2:Mb\}$

Draft/Ideas

update Library if Library HEAD has been changed something should tell the User that an update is possible. It should then (on request) change the versions of all Modules which are not in conflict with updating Library. In essence, if

$U_a:La.1 \supset \{U_a:La.1:Ma, U_b:La.2:Mb\}$ is a Library to be updated and
 $La \Rightarrow U_c:La.3 \supset \{U_b:La.1:Ma, U_c:La.3:Mb, U_c:La.1:Mc\}$ is current HEAD, then
 $U_b:La.2:Mb$ should be updated to $U_c:La.3:Mb$ and $U_c:La.1:Mc$ should be added.
 User should receive a notification that $U_a:La.1:Ma$ is not in sync with HEAD.

forking Consider forcing users to create their copy of a Package before entering to edit mode (as in *github*), find a better name if needed ...

revision graphs should be created inside this documentation.

Consider using tikz <http://www.texample.net/tikz/examples/>

To be continued...