Open XAL Versioning

Introduction

This document establishes a versioning and release policy for the Open XAL project as well as the repository layout.

Version Control

The Open XAL project will use Git for source code version control in the openxal repository of the xaldev project on Source Forge. All code and resources committed to the Open XAL project must comply with copyright laws and be consistent with the terms of our open source BSD license.

Master Branch

The main thread of development will take place on the “master” branch. All code should conform to our coding standards before being pushed to this branch on the server. One person will be responsible for managing merges to the master branch to reduce code conflicts and to enforce coding standards.

Milestone Development

Development of tasks is managed using the project’s ticket system which groups tickets among milestones. Each milestone has a specified delivery date for all the tickets within that milestone. The ticket owner is responsible for delivering the associated code for that milestone by the deadline.

Each ticket should have an associated branch in the repository with the name of the form *milestone.ticket* where the milestone is of the form “m” followed by the milestone number (e.g. “m1”) for milestone 1 and ticket is a short name describing the ticket. An example branch name is, “m1.portxal” meaning milestone 1 ticket for porting XAL code.

The owner of a ticket should manage (including delegation of responsibility) the commits for the ticket’s branch. When the branch is ready for merging into master, the ticket owner must contact the manager of the master branch. The manager of the master branch or their delegate is responsible for all merges into the master branch.

Release Management

Feature releases will be labeled using dot notation with the “release” prefix followed by the *major.minor* numeric label such as “release.2.1” for the version 2.1 release. The major number increments when the new API is no longer compatible with the previous version’s API. The minor number increments when a new feature is added, removed or modified. The first official release will begin with version 1.0. Each such feature release will be maintained in a branch named with its corresponding release label.

Patch releases will be similarly labeled using dot notation with the *major.minor.patch* numbering convention such as “release.2.1.4” for the version 2.1.4 release. The major and minor numbers correspond to the feature release. The patch number increments whenever a bug is addressed. Patch releases will be tagged in the repository with its corresponding label and continue to reside on its feature release branch.

Release Number	When to Increment
Major	Referencing APIs break or major structural changes
Minor	New features that don’t break referencing APIs
Patch	Bug fixes for existing features

Both feature and patch releases should undergo quality control review before being officially released.

Site Customization

The source code which is maintained in the master branch of the official repository on the server will be written to be commonly used among different sites. However, site customization will be necessary for such things as custom applications, device types, device properties and database queries. Each site is expected to create site specific branches off of the master and feature branches and merge against the common branches as needed to pull in the latest common code. The sites may opt to share their site specific code with the community by publishing site specific branches to the server. Site branches should be named using dot notation with the “site” prefix followed by the site name and the version. For example, the SNS master branch could be shared on the official repository with a remote branch named “site.sns.master” to indicate the SNS master branch.

Scratch Branches

Other branches which are pushed to the server may be created for developing and sharing code not ready for release. Scratch branches should be named using dot notation with the “scratch” prefix followed by a name indicating the purpose of the branch. For example, a branch named “scratch.solver” could be used for developing new solver code. Scratch branches are intended to be temporary and should be removed when no longer used.

Document Revision History

This table describes the changes to this Open XAL Versioning document.

Date	Notes
June 4, 2010	Draft proposal that describes the Open XAL versioning system.
October 21, 2010	Update the version control system to be Git instead of Subversion. This change also results in changes to the directory structure since Subversion stores branches in the main directory tree and Git does not.
September 8, 2011	Update the directory layout and introduce a new strategy for managing releases using branches and tags. Describe rules for sharing site specific and scratch code.
March 15, 2012	Update the branch naming convention to use dot notation and drop the use of separate repositories.
February 26, 2013	Added a section on milestone versioning.