Old Release Process

Introduction

About this Page

This document outlines our process for coordinating releases of the Fluid source code and the Design Handbook. This page is currently out of date.

What's in a Release?

We expect that each Fluid release will contain a combination of framework code, reusable user interface components, and documentation. More specifically:

Technology

- JavaScript code for the Fluid framework
- JavaScript, HTML, and CSS for Fluid components
- Examples and sample code
- API documentation and tutorials

Design Handbook

- UI Design Patterns
- UX Walkthrough checklists and techniques
- User testing guidelines

On this Page

Frequency of Releases

We release versions of the Fluid framework, components, and Design Handbook on a monthly basis. For more information about the contents of each monthly release, check out the Fluid Community Roadmaps.

Release Version Number

Each release has a unique version number associated with it, e.g. "0.1" or "0.3beta1". This version number must be recorded consistently in a few locations:

- Wiki pages and other documentation
- The project `pom.xml` file
- The `ant` build scripts properties file, `build-scripts/build.properties`
- The version number of the Wiki API page snapshots
  The following instructions will describe more specifically when, where and how to record the release number.

Process

As we approach an upcoming release, the following process kicks into gear:

Task	Who Coordinates with the Community?
Identify the Release Manager	Project Manager, Tech Lead

Set the release date and code freeze date	Project Manager, Tech Lead, Release Manager
Coordinate the release deliverables	Project Manager, Tech Lead , UX Leads
Work with component design/development teams to produce a test plan for each Fluid component	QA Lead
Recruit QA testers	QA Lead
Create the Release Status page	Release Manager
Update Wiki pages	Tech Lead, UX Leads
Ensure known issues in JIRA have been marked with the correct fix version for the release	Release Manager
Discuss ongoing bug fixes and commits on fluid-work	Whole Community
Ongoing QA testing and bug fixing	Whole Community
Review commits	Release Manager
Create snapshots of the wiki API documents	Release Manager
Tag and package the release	Release Manager

This is a collaborative process, and the community is encouraged to take an active role in defining schedules and coordinating the release process. It is expected that the Release Manager and QA Lead roles can be rotating positions based on interest and expertise.

About the Release Manager

The Release Manager is a volunteer from the community who agrees to be the primary point of contact for a release. The release manager's responsibilities includes:

- Working with the technical lead and project manager to determine the timing of a release
- Announcing the release schedule
- Coordinating code freeze
    - Reviewing commits
    - Ensuring post-freeze commits are well-tested, filed against known JIRA bugs, and don't change public APIs
- Managing the mechanics of a release, including:
    - Tagging the release
    - Creating a maintenance branch if necessary
    - Packaging up the release
    - Working with the project manager to announce the release

Release Status

Each Fluid release will have a status page, documenting the deliverables expected for the release. This includes a a list of new functionality, documentation, and the contents of the Design Handbook. The status page will also outline all of the known bugs and issues that are expected to be fixed in time for the release. The release status page should also include a summary of the goals for the release.

How to Create the Release Status Page

The following summarizes the steps to create a release status page:

1. In Confluence, create a new wiki page as a child of the Project Coordination page called "Fluid x.y Release Status" (where x.y is the version number).
    - This can be done by copying a previous release status page, and tweaking it.
2. Document the release goals
3. Create a table outlining the release deliverables, including the status and coordinator for each deliverable
4. In JIRA, create a filter showing all of the open issues corresponding to the release.
5. Save the filter and share it publicly. This may require special access in JIRA, so ask if you need help.
6. Grab the URL to the RSS feed of the filter.
7. Use the *jiraissues* tag in Confluence to automatically pull in the contents of the RSS feed and display it in a table:

```
{jiraissues:http://issues.fluidproject.org/sr/jira.issueviews:searchrequest-xml/temp/SearchRequest.xml?
&&pid=10001
&resolution=-1&fixfor=10000&sorter/field=priority&sorter/order=DESC&tempMax=100&reset=true&decorator=none
|columns=key;priority;summary;updated;assignee;resolution;status}
```

Updating the Wiki for Release

Updating Wiki: Development

1. Update any development API and integration pages.
2. Update any demos
3. House cleaning - Delete any unnecessary pages and information created in this past iteration.
4. Update download & component pages (see below)

Release download pages

1. Consider putting a disclaimer at the top of any affected pages, with the following text (or something similar):
    - "This page is currently being edited in preparation for the pending X.X release. It's contents should be considered in flux and unreliable until this warning is removed. For the latest stable release, see Fluid Infusion X.X."
2. Duplicate the old Fluid Infusion - Current Release page into a new page called "Fluid Infusion X.X" where X.X is the old release number.
    - _This new page should be a child of Fluid Infusion - Current Release.
3. Update Fluid Infusion - Current Release to the latest release.
    - *Be mindful to follow the excerpt format*. The links to the bundles at the top will be excerpted, and displayed on the Downloads and Demos page.
4. Update the Downloads page to reflect any new demos that are now available.

Component pages

1. Go to the Components page
2. For each Component that was updated in this release, ensure that the relevant information page is updated.
3. Update API, Integration, Demo, and Testing sections as necessary.

Updating Wiki: UX

1. House cleaning - Delete any unnecessary design information (pages, notes, etc) added to the Wiki in this past iteration.
2. Update any Components pages to reflect the current state of the design.
    - Wireframes, Storyboards, Design pattern, Functional Req, User Testing, Story Cards, User modeling, Accessibility.
3. Verify that any changes to the Design Handbook is reflected.

How to Create a Snapshot of the Wiki API Pages

The following process should be carried out for two sets of technical documentation: All Components API Documentation and Tutorials:

1. Start with the 'trunk' version of the first page in the set of documentation.
2. Create a copy of the page.
    - In the left hand menu, choose "Info"
    - On the Info page, choose "Copy"
3. Rename the copy to include the version number, using the following convention:
    - "Inline Edit API - v0.4"
4. Adjust the parent document to be the appropriate already-versioned parent.
    - Right under the title you edited in the previous step, click the yellow-highlighted "EDIT" next to the Location
    - Type in or search for the correct new parent page
    - Click "Done"
5. Edit the `{info}` block at the top of the page to declare that this page refers to the versioned doc, and they should see the other docs for trunk docs, using the following template text:
    - This documentation refers to the API documents for the v0.4 released version of the Fluid Infusion code. For documentation specific to the trunk version, please see <put a link to the trunk version of the doc here>.
6. Check the page for any links to other version-specific pages, update the links.
7. Preview the page to make sure it's ok.
8. Save the page
9. Return to the trunk version that you just created a copy of.
10. Edit the `{info}` block at the top of the trunk page to refer to the new versioned page you just created (instead of the previous version that was there).

When all the pages have been snapshotted, edit the Table of Content page to reference the new versioned docs.

How to Tag and Package the Source Code

A major constituent of the release is the "source code". This is a collection of JavaScript, Java, html, css, build script, example, and unit test files that are tracked using the fluid source code repository. The main tasks to create a release of the source code are to tag the current revisions of the source files, and to bundle them into an archive (e.g., a zip file).

How to Tag the Source Code

Note that the Fluid-0.1 release is used as an example in what follows. If you are using these instructions for another release, remember to substitute the correct version number for occurrences of "0.1".

The steps to tag the source code are:

1. Impose a source code repository freeze until the source code is tagged. Announce the freeze on the fluid-work mailing list (fluid-work@fluidproject.org).
    - Users are prohibited from *committing* to the source code repository trunk while the tagging operation is under way.
2. Modify the release information in the necessary files:
    a. Edit the maven project file(s) (`pom.xml` and `project.xml`).
    b. Set the contents of the `<version>` tags as appropriate, for example:

```
<version>0.1</version>
```

    c. Edit the ant properties file(s) (`build-scripts/build.properties`).
    d. Set the contents of the `<version>` tag as appropriate, for example:

```
fluid_version = 0.1
```

    e. Ensure that the dependencies lists in `build-scripts/build.properties` are up-to-date:
        - The dependencies lists must include all of the file that are to be included in the bundle.
        - The order of occurrance of the files in each list must take into account any dependencies between the files.
    f. Update the `README.txt` file at the root of the package to reflect the current release information.
    g. Commit these project files to the repository.
3. Check out a fresh, clean working copy of the source code from the trunk.
    - Execute:

```
svn co https://source.fluidproject.org/svn/fluid/components/trunk fluid-0.1
```

4. Using this working copy, ensure that the build works.
    a. In the `fluid-0.1` folder, execute:

```
mvn clean install
```

    b. The build should create a war file containing the fluid components. The war file is created in `.../fluid-0.1/target/fluid-components-0.1.war`, and is copied to the local maven 2 repository, `.../.m2/repository/org/fluidproject/fluid-components/0.1/fluid-components-0.1.war`.
5. Run the jqUnit tests (by opening the HTML test files in a browser), and ensure that all succeed.
6. Tag the source with the release version number.
    - Execute:

```
svn copy fluid-0.1 https://source.fluidproject.org/svn/fluid/components/tags/fluid-0.1
```

    - Note that the above assumes that "fluid-0.1" is the directory containing the maven project file with the correct release version tag.
7. Modify the version of the maven and ant project files (`pom.xml`, `project.xml` and `build-scripts/build.properties`) on trunk to reflect that trunk development is now a snapshot of the *next* release version. For example:

```
<version>0.2-SNAPSHOT</version>
```

8. Modify the fluid.version in Fluid.js
9. Update the fluid version in all the javascript files
10. Commit the trunk's modifications
11. Lift the prohibition of committing to the trunk of the repository by announcing same on the fluid-work mailing list (fluid-work@fluidproject.org).

IMPORTANT NOTE: Once the release is complete, the Image Gallery project file, found at https://source.fluidproject.org/svn/fluid/image-gallery/trunk/web/pom.xml, must be updated to reflect the new SNAPSHOT version in it's Fluid dependency.

Find the section of the `pom.xml` file that declares the dependency:

```
<dependency>
  <groupId>org.fluidproject</groupId>
  <artifactId>fluid-components</artifactId>
  <version>0.6-SNAPSHOT</version>
  <type>war</type>
</dependency>
```

and update the version to reflect the new version now in Fluid's trunk.

How to Package the Source Code

The steps to package the source code are:

1. Check out a fresh, clean copy of the *tagged* version of the source code.
    - Execute:

```
svn co https://source.fluidproject.org/svn/fluid/components/tags/fluid-0.1 fluid-0.1
```

2. `cd` into the build scripts folder (`fluid-0.1/build-scripts`) and run the `ant` task to build the release bundle:

```
ant
```

   This creates a folder called `products` and places the release bundles there: `fluid-0.1.zip fluid-0.1-src.zip`.
    - This zip file should contain the license files, as well as source, examples, and the war file.
3. Test the distribution file thoroughly by unpacking it into a clean environment and running all tests, etc.

Posting the Distribution

The distrbution file should be made available on the main project site's Downloads page (http://fluidproject.org/index.php/downloads) and on the wiki's Fluid Infusion - Current Release page.

Updating Demos on the Fluid Project website

1. Archive the existing Demos page: create a new sub page to the Demos page and copy and paste the Demos page content there.
2. Extract a copy of the new minified source code and demos into the new releases' directory. (i.e. ~/website/docs/releases/0.999beta1)
3. update the Demos page links to reference the new sample code.
    - add any new Demos
4. Add a link from the Demos page to the Demos just archived in Step 1. Put this link under the "Past Releases" section at the bottom of the page.

Update JIRA

1. Mark the version released, moving remaining open issues to the next version.
2. Do a query for all unresolved issues that affect the previous release and release candidates. Add the new release version as an affected version for each issue in the results.

Managing Unexpected Issues

Bugs happen. When unexpected issues or problems arise, the Release Manager, Technical Lead, or Project Manager will inform the community and work with them to adjust the release schedule accordingly. If you find an issue that you think is a blocker, let the Release Manager know as soon as possible.

If you discover a security issue, follow the Fluid Security Policy and report the issue privately to the security team.