How to Release Forrest

This documents the steps that the Release Manager (RM) should follow when doing a Forrest release.

Table of contents

1. About this Document

Warning:
This document is still being developed from etc/RELEASE_PROCESS.txt and some steps will need to be re-arranged.

This documents the steps that the Release Manager (RM) should follow when doing a Forrest release. Note that it might have mistakes - we seem to discover something new each time and some steps might need to happen in a different order. Fine tune these notes for next time. Do some practice runs.

There are some steps that other committers, and even developers, can assist with, especially in the areas of getting ready for the release and the final testing. Many of the steps can be done only by the Release Manager.

It is not the Release Manager's job to fix bugs nor address blocker issues. The RM job begins when the project is ready to do the release.

2. Preparing the Project for the release

1. The Release Manager (RM) starts the process to finalise the outstanding blocker issues.

   Check and make sure the following preconditions are met:

   - Has the project prepared or updated the Roadmap to schedule the realistic Issues?
   - Has the project made good progress towards fixing the Blockers and applying the outstanding patches?

   If not, then the project is not yet ready for release. Remember that it is not the RM's job to do this.

   If so, then send an email to get the project to decide what to do with the remaining issues. Propose to delay some issues to a future release, encourage people to fix others. See FOR-853. Look at msg02310.html for an example of such a message.
2. Start discussion on Java-Version to use for compiling and testing the release.

3. Preparations for the Release Team

Particularly the Release Manager, but also anyone assisting, needs to be familiar with standard procedures and Apache terminology. This is crucial for a successful release.

1. If you have never done a release before or need to refresh your memory, read all about Apache Releases in general at http://www.apache.org/dev/#releases. Make sure any assistants have read and understood this as well.
2. Make sure every team member is familiar with the process of signing releases and generating MD5 and PGP. You'll find some more info at Signing Releases> and http://forrest.apache.org/mirrors.cgi#verify
3. Ensure that as many PMC members as possible have their PGP keys in the KEYS file in Forrest's root directory. Instructions on how to add keys are included in that file. Instructions on how to create and manage pgp-keys can be found at the abovementioned references.
4. Make sure every team member has downloaded and installed the Java-Version to use for compiling the release. Downloading and installing that version should be done well ahead of time to avoid

delays.

4. Prepare the Release Plan

Prepare the Release Plan to define the corner stones of the coming release

1. Java-Version to test this release
2. Start of code-freeze
3. Start of test-period
4. Vote on release candidate
5. Optional creation of release candidate #2 (when there are bugs)
6. Start of test-period #2
7. Vote on release candidate #2
8. Scheduled release Date

Use the email template propose_release_plan.txt to write and propose your plan, then call for a quick vote on the release plan on the dev list.

Note:

There are various reasons for voting on the Release Plan, e.g. makes people aware that a code-freeze is about to happen; encourage them to get involved with the release; ensure that the date is suitable and people will be around to test and then vote on the actual release. See a good discussion in the archives

5. Preparing the Code Base

1. Ensure that there are no copyright issues. The committers and PMC would have been continually monitoring this. There are some tools to assist with scanning for issues, e.g. `svn:committers/relicense/src/perl/relicense.txt` and svn:committers/tools/
2. Ensure that the line-endings and svn:eol-style property are correct for all files. See svn:committers/tools/
3. Ensure that documentation is ready.
4. Ensure that all relevant plugins have been deployed to plugins/0.8-dev/ See other notes at plugins/RELEASE_PROCESS.txt

FIXME (fso):

Check and integrate plugins/RELEASE_PROCESS.txt as a new document.

6. Prepare Release Branch

FIXME (fso):

We need to discuss order from here on. My idea is to adjust docs before we enter code freeze to save time later. But if the rc fails and release is postponed might need to roll back changes easily and - if possible - roll them forward later. So creating an svn branch for the rc seems to make sense to me. Also: For more than one person working on building the release for different OS it would also be good to have all the changes in svn committed already rather than doing it later. Probably easiest would be to create an rc branch here and co that. I'd sacrifice the alternative approach for that which is far too risky for my liking anyway. wdyt?

In this step you check out a fresh copy from SVN to make sure you have no local modifications, especially those that might be hidden by svn:ignore settings. It will soon become the release branch.

Note:
This step is actually just preparation to keep code-freeze period as short as possible.

FIXME (dc):
What does that mean?

1. Create a new empty directory 'Forrest_Release' on your system and make it the current directory.
2. Start `svn co https://svn.apache.org/repos/asf/forrest/trunk` from the command-line of your system or the equivalent for the svn-tool you use.

Note:
This will take quite a while if you are on a dial-up connection. See alternatives below.

Alternative Approach

1. Do 'svn update -r HEAD' to ensure that you are up-to-date.
2. Run 'svn status --no-ignore'
3. Delete any extra files you might have added/changed in your local copy. **They must not be packed with the release.** It must be a pristine copy of the current trunk.

Warning:
This approach requires a good understanding of svn and how it works. It is not as automatic and safe as the method above.

7. Prepare Docs for next release cycle

FIXME ():
I'd suggest the following steps to keep build size small and simplify procedure:

1. Edit version subtabs in site.xml as follows:
   1. Move all version numbers one line down so that

```
<versions tab="docs">
   <overview label="Overview" href="versions/index.html"/>
   <v0.8 label="0.8-dev" href="site:v0.80//index"/>
   <v0.7 label="0.7 (current)" href="site:v0.70//index"/>
   <v0.6 label="0.6" href="site:v0.60//index"/>
 </versions>
```

   becomes

```
<versions tab="docs">
   <overview label="Overview" href="versions/index.html"/>
   <v0.9 label="0.9-dev" href="site:v0.90//index"/>
   <v0.8 label="0.8 (current)" href="site:v0.80//index"/>
   <v0.7 label="0.7" href="site:v0.70//index"/>
 </versions>
```

2. Remove past versions (0.6) docs-directory from svn branch.

FIXME (fso):
find and list svn-command

3. Adjust version-numbers in site.xml.

FIXME (fso):
This used to be 'Do global replace throughout docs_0_80 to replace the string ="site:v0.70 with ="site:v0.80' but this needs checking.

4. Edit site-author/status.xml:
   1. Remove the -dev from the current <release> tag, and set the release date.
   2. Add a new <release> for development on the next version e.g. from: <release version="0.7-dev" date="not yet released"> ... to: <release version="0.8-dev" date="not yet released"> </release> <release version="0.7" date="2002-02-13"> ...
5. Edit the forrest/site-author/content/xdocs/mirrors.html and adjust all version-specific content.

FIXME ():
FIXME: There is a bug (FOR-300) in the forrest build which generates to main/site/mirrors.html instead of build/site/mirrors.html

6. Edit the Forrest home page in the "News and events" section and add a text like:

```
Apache Forrest 0.xx was released on [Date]. [Important new features]
```

7. Rename the deployed plugins directory by issuing the following commands at the command line

```
cd /svn/asf/forrest-site
svn update
svn mv plugins/0.8-dev plugins/0.8
svn mkdir plugins/0.9-dev
svn status
svn commit
```

FIXME (fso):
Issue them where and to what end?

8. Building the distribution

In this phase you build the release candidate to be tested.

Note:
You can practice the following steps (as far as creating the branch) without committing anything even before code-freeze. This ensures a good release candidate.

1. Use template anounce_code_freeze.txt to send email to dev-list that the code-freeze has now commenced.
2. Update your release checkout (svn up) to include last minute changes.
3. Run the following quick tests from the command line of your system to ensure that all is well:
   - Change to the main directory and run `build test`. The build should conclude without errors.
   - Change to the site-author-directory and run 'forrest'. The docs should build without errors.

   If there are any problems, focus on problems that prevent building and invite other commiters to help you solve the problems.

Note:
It is not your job to fix bugs and code freeze should not commence with a broken trunk.

If there are bugs that cannot be easily fixed, then call a halt to the release process and start a discussion on rescheduling options on the dev-list with the template rc_did_not_build_what_now.txt

4. Remove the build directories from core and plugins. Do `svn st --no-ignore` in the root directory of your release candidate directory to be sure that all files created by the test build have been removed and no other files have been changed. The status command should report no changes.
5. Update the version numbers at various places:
   - Edit main/build.xml and replace the '-dev' text with '' i.e. nothing: around line 45: <property name="forrest.version" value="0.7-dev"/> to: <property name="forrest.version" value="0.7"/>
   - Edit main/forrest.build.xml to update the version tag to remove "-dev". There are two occurences: around line 32: <property name="forrest.version" value="0.7-dev"/> ^^^^ around line 60: <description> | Forrest Site Builder | | 0.7-dev | ^^^^
   - Edit plugins/build.xml and increase the docs version number to the next major release: around line 23: <property name="forrest.version" value="0.7"/> to: <property name="forrest.version" value="0.8"/>

Note:
This is deliberately a major version up. It is assumed that plugins will be developed against the next version of Forrest. Individual plugins can override this property in their own build files.

6. Ensure that each plugin that uses the locationmap has its "release version" set to 0.8 or more.
7. Edit 4 files in tools/forrestbar to update the version number to match the new release: - install.rdf, line 24: <em:version>0.7</em:version> - install.js, line 19: var err = initInstall("ForrestBar", "forrestbar", "0.7"); - xpi/chrome/content/contents.rdf, line 27: chrome:displayName="ForrestBar 0.7"/> - xpi/chrome/content/forrestbarOverlay.xul, about line 40 edit the version number as well as change the link to point to the new release's docs: <menuitem label="Current Docs (0.7)" onclick="navigate('http://forrest.apache.org/docs_0_70/index.html');" />

FIXME ():
There are probably other areas which have version numbers. How can we improve this?

FIXME ():
Not sure at what stage we get rid of the old docs, e.g. 0.6

FIXME ():
Not sure at what stage need to edit site-author/content/xdocs/mirrors.html (Presume that it should be done after packing release. See below.)

8. Create a new file, etc/RELEASE-NOTES-x.y.txt, where x.y is the version currently being released. It is best to copy an earlier RELEASE-NOTES file, to keep a common layout. In this file, provide a summary of changes, and check for general accuracy. Scan the status.xml/changes and the Roadmap via the issues tracker, to find the important issues.
9. Set your Java version to be the lowest specified of our supported versions.

Note:
Set the environment variable JAVA_HOME to the path of the Java version. Note for Windows: If you change the setting in system properties, you need to logout and login again for the changes to become effective.

10. Take note of the SVN revision number of your trunk by running `svn info` from the command line in the Release Candidates root dir and look at the "Last Changed Rev: ######".

FIXME ():
What is this used for?

11. Now we will build the release candidates for Windows and Unix.

Note:
The reason for creating two separate archives is the line-endings dilemma between Windows and UNIX. SVN ensures correct line-endings on each operating system (as long as committers have been diligent when adding/updating the repository).

- On a UNIX machine:
  Change to directory main and run `build release-dist` to generate the distributions on a UNIX machine.

  Two archives are created: apache-forrest-X.Y.tar.gz apache-forrest-X.Y.zip. Ignore the *.zip archive.

  Unpack and test the relevant archive in a fresh new directory.
- On a Windows machine:
  Change to directory main and run `build release-dist` to generate the distributions on a UNIX machine.

  Two archives are created: apache-forrest-X.Y.tar.gz apache-forrest-X.Y.zip. Ignore the *.tar.gz archive.

  Unpack and test the relevant archive in a fresh new directory.

12. Sign the Release Candidates distribution file and the *.asc and *.md5 files.

    Here is one example when using gpg and openssl from the command line.

Note:
An windows version for openssl can be found at http://www.slproweb.com/products/Win32OpenSSL.html

```
gpg --recv-key <myKey>
gpg --output crossley-apache-forrest-0.7-RC1.tar.gz.asc \
--detach-sig --armor apache-forrest-0.7-RC1.tar.gz
gpg --verify crossley-apache-forrest-0.7.tar.gz.asc \
apache-forrest-0.7-RC1.tar.gz
```

... should say "Good signature from ..."

```
openssl dgst -md5 -out apache-forrest-0.7.tar.gz.md5 \
apache-forrest-0.7-RC1.tar.gz
md5sum apache-forrest-0.7-RC1.tar.gz
```

... output should match that of the md5 file.

13. Create a maintenance branch in SVN

1. Open the command line
2. Change to the root directory of the release candidate
3. run `svn copy -m "Create the x.y release branch from r#####" \ https://svn.apache.org/repos/asf/forrest/trunk \ https://svn.apache.org/repos/asf/forrest/branches/forrest_xy_branch` where 'xy' is a compact form of the version (e.g. 04, 041, 05) and 'r#####' is the SVN revision number that the branch was created from which was the revision that the release candidates were generated from.

   See http://svn.apache.org/repos/asf/forrest/branches/ If someone has done a commit before you get to do it, then specify the revision number with -r

FIXME ():

What do I see at http://svn.apache.org/repos/asf/forrest/branches/ if s.o. has done a commit? What is this stuff revision numer with -r for?

9. Testing the release candidate

Test the actual distribution on various platforms.

1. Upload the release candidates and signatures to a committer's webspace. Use the .tar.gz from the UNIX machine and .zip from the Windows machine.
2. Use template test_and_vote_on_rel_cand.txt for an email to the dev-list asking all developers to test and vote.
3. As the votes come in
   - Make sure the distributions unpacks on different systems w/o problems.
   - Make sure that somebody has followed the actual user instructions in the Forrest distribution at README.txt and index.html
   - Encourage people to build ome difficult sites.
4. If necessarry start again with Building the distribution and build another release candidate.
5.

10. Finalizing the Release

When a good release candidate has been achieved and affirmed by the vote, we'll finalize the release.

1. rename the Release Candidates distribution files apache-forrest-X.Y-RCx.tar.gz and apache-forrest-X.Y-RCx.zip to their final filenames apache-forrest-X.Y.tar.gz and apache-forrest-X.Y.zip
2. Create new .md5 and .asc-files following the procedure in outlined above
3. If there have been changes to the trunk since the branch was created, then merge trunk to branch.

FIXME (fso):

What is the purpose of this step? It doesn't seem to be right because trunk may already contain parts of the next version. What we should do is do all fixing of RC-problems in the rc-branch (same as changing docs) then, on release, merge branch back into trunk to integrate fixes and doc-changes back into trunk. wdyt?

4. If everything looks okay tag SVN by running `svn copy -m "Create tag forrest_xy from release branch" \ https://svn.apache.org/repos/asf/forrest/branches/forrest_xy_branch \ https://svn.apache.org/repos/asf/forrest/tags/forrest_xy` from the

command line of your system, where 'xy' is a compact (without the dots) form of the version number (e.g. 04, 041, 05).

FIXME (fso):
If we change procedure to create an rc-branch this will become a merge changes from trunk then rename rc-branch to final release branch. right?

See http://svn.apache.org/repos/asf/forrest/tags/ for more information.

FIXME (fso):
What if it doesn't, how do I tell, what do I do?

5. Announce the end of the code-freeze by sendung the email-template announce_end_of_code_freeze.txtto the dev list.

11. Upload and announcement

In this phase we'll upload the new Release, wait for it to be available on most mirror sites, then announce the new release.

Note:
During this phase there is a lot of waiting. While things are happening you can be doing the Cleanups described below.

1. Use scp to upload the release: the *.tar.gz, the *.zip, the *.asc and *.md5 files, and the RELEASE-NOTES-x.y.txt to people.apache.org at /www/www.apache.org/ dist/forrest/

   Ensure correct file permissions by executing `chgrp forrest *` then `chmod 664 *` on the remote system.

   Each PMC member has a server account and belongs to the forrest group.

   The process is documented at http://www.apache.org/~bodewig/mirror.html and http://www.apache.org/dev/#releases

   Leave the previous dist there as well as the new one, until after the announcement (because mirrors.html cannot be updated until most mirrors have received the release).

Note:
The other files there (HEAD.html README.html LICENSE.txt KEYS) are all automatically updated from the SVN:forrest/dist/ repository.

FIXME ():
FIXME: Add notes about the KEYS file in the "forrest-dist" SVN respository.

2. Wait for the various mirrors to pick up the new files.

   For some mirrors, this takes only a few hours. However others are slow. How long to wait is a tradeoff, e.g. 8 hours.

   See Status of mirrors.

   Take note of the time that the eu.apache.org mirror is updated, then compare each "mirror age" to that.

When you see that a good proportion of the mirrors have received the release, then update the website, then send the announcement.

3. Create a copy of current dev-docs in trunk for the next development phase. Do 'cd site-author/content/xdocs' and 'svn copy docs_0_70 docs_0_80' (Adjust version numbers as needed).
4. Open site.xml and add a copy of the most current versioned section (e.g. <v0.80>) above it. Increment the first decimal of the sections name to reflect the next planned release (e.g. <v0.90>).
5. Update the .htaccess file to redirect /docs/dev/ to the next version, and do other changes noted in the .htaccess file. See site-author/content/.htaccess

FIXME (fso):

Need to go through .htaccess and clean up.

6. Rebuild (Forrest site) and publish the Forrest website as normal. Be sure to use the new version for building the docs. Refer to Publishing Forrest Documentation for details.
7. Update the xml.apache.org website (Forrest is part of the Apache XML federation of projects). Edit xml-site/src/documentation/content/xdocs/news.xml and record the announcement, and then commit the new HTML to xml-site/targets/forrest Note that they use forrest-0.7 to build their website. See http://xml.apache.org/guidelines.html#website-top
8. Send announce_release.txtas email to 'dev@forrest.apache.org', 'user@forrest.apache.org', 'announce@apache.org', 'announcements@xml.apache.org'. Sign the email (e.g. PGP).

   See previous announcements for examples:

   - 0.2
   - 0.3
   - 0.4
   - 0.5
   - 0.5.1
   - 0.6
   - 0.7

9. Do the Freshmeat announcement: http://freshmeat.net/projects/forrest/

12. Cleanup

1. Edit main/build.xml, increment the version and add a -dev tag: around line 45: <property name="version" value="0.8-dev"/>
2. Edit main/forrest.build.xml and update the version: around line 32:

```
<property name="version" value="0.8-dev"/>

around line 52:
<description>
|                  Forrest Site Builder                  |
|                         0.8-dev                        |
```

3. Remove old dist files from the /www/www.apache.org/dist/forrest/ directory. They have already been automatically archived at archive.apache.org/dist/forrest/
4. Do some Jira administration (need to be in the jira-administrators group)

FIXME (fso):

Does it make sense to pass this job to the Jira-role?

1. Tweak the "release" versions via "admin" interface at our Jira. Do this ...
2. Re-name the VersionIds using "Manage Versions" then "Edit details": e.g. 0.7-dev is renamed to 0.7 and 0.8 becomes 0.8-dev
3. Mark 0.7 as released using "Manage Versions".
4. Review the Issues for the old version and move any Incomplete ones up.
5. Change the "fixfor" attribute to the next verion for the "project.issues-rss-url" RSS feed in forrest.properties

5. Cleanup this RELEASE_PROCESS.txt file to set version number examples to be ready for the next release.

FIXME (fso):

I'd like to drop this step and rather word everything more flexibly.

6. Remove the release candidates from your public_html directory.

13. Conclusion

All done!

Or perhaps not.. if you think of anything, please refine these instructions.