Geronimo Component Release Process

Change Policy

Everyone is encouraged to update this documentation with clarifications, use of newer maven tooling, etc. Only major changes inconsistent with the spirit of this process need to be discussed on the dev list.

Procedure

1. Whenever possible, use the maven release plugin. If something doesn't work file a bug against it.
2. Use extreme caution in creating branches as opposed to releasing from trunk. While "core" geronimo may need to keep branches, most smaller projects such as specs, plugins, components, and most likely tools should avoid the complexity of branches unless clearly necessary and agreed upon.
3. When branches are needed, branches/x.y would be the branch for all x.y.z releases

The next sections are copied from http://maven.apache.org/developers/release/releasing.html with modifications for Geronimo.

Releasing A Geronimo Project

What follows is a description of releasing a Geronimo project to a staging repository, whereupon it is scrutinized by the community, approved, and transfered to a production repository.
Prerequisite

Be sure that:

- you have all Maven servers defined in your `settings.xml`. For more information, please refer to Maven Committer settings which also apply for Geronimo committers.
- you have created your GPG keys. For more information, please refer to Making GPG Keys.

In order to release a project you must also have the following setup in your `$HOME/.m2/settings.xml` which is a profile that defines the staging repository.

Here's what your release profile might look like in your `$HOME/.m2/settings.xml` :

```
<settings>
    <profiles>
        <profile>
            <id>release</id>
            <properties>
                <gpg.passphrase>[secretPhrase]</gpg.passphrase>
                <deploy.altRepository>apache.releases::default::scp://people.apache.org/x1/home/[your apache id]
/public_html/staging-repo/${siteId}</deploy.altRepository>
                <staging.siteURL>scp://people.apache.org/x1/home/[your apache id]/public_html/staging-site<
/staging.siteURL>
           </properties>
        </profile>
        <profile>
            <!-- use for local site deploy testing and local deploy testing -->
            <id>local</id>
            <properties>
                <deploy.altRepository>djencks::default::file://[home directory]/staging-repo/${siteId}</deploy.
altRepository>
                <gpg.passphrase>[secretPhrase]</gpg.passphrase>
                <staging.siteURL>file://[home directory]/staging-site</staging.siteURL>
            </properties>
        </profile>
    </profiles>

    <servers>
        <server>
            <id>apache.releases</id>
            <username>[your apache id]</username>
            <passphrase>[secret passphrase]</passphrase>
            <filePermissions>664</filePermissions>
            <directoryPermissions>775</directoryPermissions>
        </server>
        <server>
            <id>geronimo-website</id>
            <username>[your apache id]</username>
            <passphrase>[secret passphrase]</passphrase>
            <filePermissions>664</filePermissions>
            <directoryPermissions>775</directoryPermissions>
        </server>
    </servers>
</settings>
```

The server name apache.releases at the start of deploy.altRepository must correspond to the apache.releases server definition.
Also that your apache id does not start with "~".

Everything that you need to release has (will have, actually) been configured in the genesis root pom all Geronimo projects inherit from.

Your project should adhere to standard trunk,branches,tags svn layout in which case no further release profile configuration should be needed. Some slight deviation such as our specs project still works without extra configuration. Avoid more complex layouts that require special configuration.

This is the base release configuration in the genesis root pom:

```
        <profile>
            <id>release</id>

            <build>
                <plugins>

                    <plugin>
                        <groupId>org.apache.maven.plugins</groupId>
                        <artifactId>maven-release-plugin</artifactId>
                        <configuration>
                            <useReleaseProfile>false</useReleaseProfile>
                            <goals>deploy</goals>
                            <arguments>-Prelease</arguments>
                        </configuration>
                    </plugin>
```

```
                <!-- We want a source jar -->
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-source-plugin</artifactId>
                    <executions>
                        <execution>
                            <goals>
                                <goal>jar</goal>
                            </goals>
                        </execution>
                    </executions>
                </plugin>

                <!-- We want to sign the artifact, the POM, and all attached artifacts -->
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-gpg-plugin</artifactId>
                    <inherited>true</inherited>
                    <configuration>
                        <passphrase>${gpg.passphrase}</passphrase>
                    </configuration>
                    <executions>
                        <execution>
                            <goals>
                                <goal>sign</goal>
                            </goals>
                        </execution>
                    </executions>
                </plugin>

                <!-- We want to deploy the artifact to a staging location for perusal -->
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-deploy-plugin</artifactId>
                    <inherited>true</inherited>
                    <configuration>
                        <altDeploymentRepository>${deploy.altRepository}</altDeploymentRepository>
                        <updateReleaseInfo>true</updateReleaseInfo>
                    </configuration>
                </plugin>

                <!-- We want the JavaDoc JAR published with the release -->
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-javadoc-plugin</artifactId>
                    <inherited>true</inherited>
                    <configuration>
                        <source>1.5</source>
                    </configuration>
                    <executions>
                        <execution>
                            <id>attach-javadocs</id>
                            <goals>
                                <goal>jar</goal>
                            </goals>
                        </execution>
                    </executions>
                </plugin>
            </plugins>
        </build>
    </profile>
```

Release Process for Part Of Geronimo

1. Prepare your poms for release:
Genesis and the recommended settings.xml file above rely on the existence of some configuration in the top level pom in your project to parameterize where in your staging area the release will be located. This makes it so you can have multiple independent releases under vote in your staging area at once, and clearly locates the maven generated site in the documentation once the release is completed. You should only need to do this the first time these instructions are used.

```
<properties>
<!-- siteId locates the staging repo and staging site via the settings.xml profiles -->
    <siteId>plugins/directory</siteId>
<!--other properties your project may need -->
</properties>

<!-- this locates the eventual maven generated site -->
<url>http://geronimo.apache.org/maven/${siteId}/${version}</url>
<!-- this locates the staging site for the maven generated site -->
<distributionManagement>
    <site>
        <id>geronimo-website</id>
        <url>${staging.siteURL}/${siteId}/${version}</url>
    </site>
</distributionManagement>
```

- Try out the release and look for anything strange such as incorrect target locations in the modified poms, missing licenses, etc.

  ```
  mvn release:prepare -Prelease -DdryRun=true
  ```

- Diff the original file `pom.xml` with the one called `pom.xml.tag` to see if the license or any other info has been removed. This has been known to happen if the starting <project> tag is not on a single line. The only things that should be different between these files are the `<version>` and `<scm>` elements. Any other changes, you must back-port yourself to the original `pom.xml` file and commit before proceeding with the release.
- Remember to do `mvn release:clean` before you start the real release process.
  2. (optional) Publish a snapshot:

```
mvn deploy
...
[INFO] [deploy:deploy]
[INFO] Retrieving previous build number from apache.snapshots
...
```

Important

IMPORTANT NOTE: Be sure that the generated artifacts respect the Apache release rules : NOTICE and LICENSE files should be present in the META-INF directory within the jar. For -sources artifacts, be sure that your pom does NOT use the maven-source-plugin:2.0.3 which is broken. The recommended version at this time is 2.0.4. All this should be automated
via the maven-remote-resources-plugin (installs legal files) and the tools-maven-plugin (checks for legal files).

If during mvn deploy command, you are challenged for password many times, you may want to create ssh private and public keys.
You could verify the deployment under Apache Snapshot repository.

```
http://people.apache.org/repo/m2-snapshot-repository/org/apache/geronimo/...
```

3. Prepare the release

```
mvn release:prepare -Prelease
```

Preparing the release will create the new tag in SVN, automatically checking in on your behalf.

getting authorization failure

If the prepare release command failed because of authorization failure, like below:
svn: MKACTIVITY of '/repos/asf/!svn/act/6ed2cc4d-dae9-4134-9cfb-f17cc8dd02ea': authorization failed (https://svn.apache.org)
You can issue "mvn release:prepare -Prelease -Dusername=username -Dpassword=password" instead.

4. Make a copy of the checked out project in this state in case you need to roll back the release

```
cd ..
cp -r trunk trunk-prepared
cd trunk
```

AFAICT mvn release:rollback only works on a checkout on which mvn release:prepare has been run but not mvn release:perform.

5. Stage the release for a vote

```
mvn release:perform -Prelease
```

If the project lists the default modules in a profile you MUST include that profile and the release profile in the command line! For instance if the default profile id is "default" the command line would be

```
mvn release:perform -Pdefault,release
```

6. Stage the latest documentation (if there is an actual maven generated site)

See the genesis project-config site page for instructions on how to set up your project so site staging works.
Note that the -Prelease profile is needed to specify the profile in settings.xml that configures the staging location.
Build the site from the tag that release:perform checked out into target/checkout in step 5.

```
cd target/checkout
mvn site site:deploy -Prelease
```

7. Propose a vote on the dev list with the closed issues, the issues left, the staging repository and the staging site. For instance:

```
To: "Geronimo Developers List" <dev@geronimo.apache.org>
Subject: [VOTE] Release Geronimo xxx version yyy

Hi,

<info about release>

Staging repo:
http://people.apache.org/~YOUR_APACHE_USERNAME/staging-repo/[siteId]...

Staging site:
http://people.apache.org/~YOUR_APACHE_USERNAME/staging-site/[siteId]...

The svn location is here:
https://svn.apache.org/repos/asf/geronimo/...

Vote open for 72 hours.

[ ] +1
[ ] +0
[ ] -1
```

Once a vote is successful, post the result to the dev list and cc the pmc.

In case the vote fails, rollback the release using the backup copy you made in step 4

```
mvn release:rollback
```

You also have to remove the tag from svn by hand.

8. Copy from the staging repo to the production repo
Once the release is deemed fit for public consumption it can be transfered to a production repository where it will be available to all users.

Note: Current version of the stage plugin is 1.0-alpha-1. There's probably an easier way, but I added it to a pom and built to pull it down.

This version of the stage plugin does not work with maven 2.0.9. Use a suitable path to maven so you run it using maven 2.0.8. Note that the stage plugin does not use any info from the project so this should cause no problems

Here is an example on how to use the stage plugin:

```
mvn stage:copy -Dsource="http://people.apache.org/<your apache id>/staging-repo/<siteId>" \
  -Dtarget="scp://people.apache.org/www/people.apache.org/repo/m2-ibiblio-rsync-repository" \
  -Dversion=2.3 \
  -DtargetRepositoryId=apache.releases
```

The version parameter is currently ignored and the entire staging repository is synced, not just the given version or the current project. It still needs to be provided, though.
Ensure that your public PGP key is in appropriate public locations (esp. /www/www.apache.org/dist/geronimo/KEYS on people and http://pgp.mit.edu/) and that your pgp key has been signed by several other apache committers to create a web of trust.

9. Deploy the current and versioned websites (if there is a reasonable maven generated site)

```
ssh [apacheId]@people.apache.org
chgrp -R geronimo public_html/staging-site/[siteId]
chmod -R g+w public_html/staging-site/[siteId]
cp -r public_html/staging-site/[siteId] /www/geronimo.apache.org/maven
chmod -R g+w /www/geronimo.apache.org/maven/[siteId]
```

10. Review Website (if any)

Wait for the files to arrive at

http://geronimo.apache.org/...

11. Update the plugins page
If this is a plugin release, update the apache `geronimo-plugins.xml`.
PROPOSAL ON HOW TO DO THIS:

First, perform the release. This process could be done as part of the release but that may be too confusing. The important point is to build the new tag with a fresh copy of geronimo-plugins.xml.
The official `geronimo-plugins.xml` is located in svn under e.g. geronimo/site/trunk/docs/plugins/geronimo-2.1/geronimo-plugins.xml
copy this file to `~/.m2/repository/geronimo-plugins.xml`
run `mvn clean install` on a fresh checkout of the new tag.
copy the merged geronimo-plugins.xml back your svn site checkout.
Put the apache license header back in to the merged file (the car-maven-plugin removes it)
Edit the source-repository elements in the new plugins if necessary. They should contain

```
            <source-repository>http://repo1.maven.org/maven2/</source-repository>
```

and most likely nothing else.
commit the svn revisions.

12. Update JIRA

Go to Admin section in JIRA and move the released version to released and make a new version.
13. Create an Announcement. For instance:

```
From: YOUR_APACHE_EMAIL
To: dev@geronimo.apache.org, users@geronimo.apache.org
Subject: [ANN] Geronimo Foo Released

The Geronimo team is pleased to announce the release of the Geronimo Foo, version Y.Y

This foo (insert short description of the foo's purpose).

Release Notes - Geronimo Foo - Version Y.Y

(Copy Here Release Notes in Text Format from Jira)

Enjoy,

-The Geronimo team
```

we don't seem to have a "next board report" or a recent releases page in the wiki

14. Add the release to the next board report, in the private subversion area.
15. Add the release to the wiki, under the Recent Releases section of the front page and on the Releases page.
16. Celebrate :o)

Notes and Gotchas

- If the selection of modules for the default build is set in a default profile then more work will be necessary. When genesis did not have the modules specified in the base pom this involved running mvn release:perform -Pdefault,release which only deploys the root, then running mvn deploy -Prelease in each of the 3 modules listed in the default profile. For projects that are not expected to be used as parents of independently releasable projects (for instance server/trunk) including the list of modules in the release profile override should work. Better still is just having the modules outside a profile.

When using the maven release plugin is impossible (this should be a less frequent event as we progress):

1. when a release is frozen, we spin off a branch with that **exact** name, as in branches/x.y.z, where z starts at zero and increments by one.
2. at that time branches/x.y is immediately updated to version x.y.(z+1)-SNAPSHOT
3. We cut releases from the frozen branch
4. When a release passes final tck testing and final vote, the frozen branch is moved to tags

Updating the poms after making a new branch

Once a new branch is created you will generally need to manage the version number in the poms. The following Perl scripts will assist in that task. It could use some polishing but given the relatively infrequent use.

Pom Version Changer

```
perl -i.orig -pe '
$done = 0 if /<?xml/;
$inParent = 1 if not $done and /<parent>/;
s,oldVersion</version>,newVersion</version>, if $inParent and not $done;
$done = $inParent = 1 if /<\/parent>/;
' $(find GeronimoDirectory -name pom.xml | grep -v "GeronimoDirectory/pom.xml")
```

Remember to properly escape periods in the ***oldVersion***. For instance, to change 1.1.1-SNAPSHOT to 1.1.1 you would have

Example

```
perl -i.orig -pe '
$done = 0 if /<?xml/;
$inParent = 1 if not $done and /<parent>/;
s,1\.1\.1-SNAPSHOT</version>,1.1.1</version>, if $inParent and not $done;
$done = $inParent = 1 if /<\/parent>/;
' $(find GeronimoDirectory -name pom.xml | grep -v "GeronimoDirectory/pom.xml")
```

making the above script work

⚠ You must replace ***GeronimoDirectory*** above with the fully qualified path to the directory (using "~" will not work).
Also note: There are references to versions outside of the pom parent entries updated by the script which will also need to be updated with the new version.

Rationale

We create a branch at freeze time for the following reasons:

1. it takes at least one week from freeze to ship due to voting, tck testing and potential repeats of that process (re-cut, re-certify, re-vote). There is no reason why work on x.y.z+1 needs to be delayed - only 52 weeks a year.
2. stronger guarantee no one is updating the branch once frozen
3. less likely that people and ci systems (continuum) will checkout and build pre-release versions of x.y.z (not x.y.z-SNAPSHOT) which would need to be removed manually and may accidentally be distributed.
4. it is currently very difficult to roll version numbers forward, entries here and there are often missed. Far better to have branches/x.y have a few straggling old x.y.z-SNAPSHOT versions than a few overlooked x.y.z final numbers that needed to go back to SNAPSHOT - they never leave SNAPSHOT and need to be reverted back later if that process happens in the frozen branch.

Steps

1. Download and install the Gnu Privacy Guard (GPG) from http://www.gnupg.org. Read the documentation on that site and create a key. Have the key signed and verified by others. Submit your public key to http://pgp.mit.edu/. This is a one time process.
2. Create a "staging" profile in your `~/.m2/settings.xml`

```
<profile>
    <id>staging</id>
    <properties>
        <!-- deploy.altRepository>prasad::default::scp://people.apache.org/x1/home/prasad/public_html/2.
0-M1-rc1</deploy.altRepository -->
        <deploy.altRepository>prasad::default::file://c:\cygwin\home\prasad\releases</deploy.
altRepository>
        <gpg.passphrase>Your GPG Passphrase</gpg.passphrase>
    </properties>
</profile>
```

3. Copy (or move as per situation, for eg specs) the trunk to branches using the following command.

```
svn mv SRC-URL DEST-URL -m "Reason for this commit".
```

4. Checkout or update this branches tree on your machine.
5. Update the <scm> urls in the `pom.xml` to point to the final url in tags. Eg:

```
<scm>
     <connection>scm:svn:http://svn.apache.org/repos/asf/geronimo/specs/tags/geronimo-servlet_2.5_spec-
1.1</connection>
     <developerConnection>scm:svn:https://svn.apache.org/repos/asf/geronimo/repos/asf/geronimo/specs/tags
/geronimo-servlet_2.5_spec-1.1</developerConnection>
     <url>http://svn.apache.org/viewvc/geronimo/repos/asf/geronimo/specs/tags/geronimo-servlet_2.5_spec-
1.1</url>
</scm>
```

6. Build the new branches tree that will soon be released using the following command.

```
Genesis 1.x -
    mvn -Pdefault,release deploy
Others -
    mvn -pdefault,staging deploy
```

Staging

Before running this step, verify that there is a corresponding "staging" or "release" profile in pom.xml. Some of the newer releases, will require you to use -Pdefault,release as we are trying to make it impossible to directly release artifacts to the apache.release repo by providing a default "release" profile that stages files to public people.apache.org directory.

7. Go the temporary staging directory specified by `deploy.altRepository` element in the *staging* profile of your `settings.xml`. Delete all *.asc.* files under this directory tree. Tar the staging directory using the command

```
find . -name *.asc.* | xargs rm -f
tar -zcvf release.tar.gz releases
```

Ensure you include the *.asc files, as these are required and will be checked for by the Apache Repository team.

8. Copy the tar ball to a publicly accessible location. Put it for a vote. In the vote notice, please include the precise names and versions being voted on (e.g. geronimo-javamail_1.4_spec-1.1) and the svn urls to the current source and where the tag will be created.
9. After it has been approved, untar the tar ball into the appropriate maven structure on people.apache.org under the directory /www/people.apache.org/repo/m2-ibiblio-rsync-repository. A cron job will rsync this with ibiblio and release it into the wild.

```
gunzip foo.tar.gz
tar -xvf foo.tar
```

Ensure that the files you copy to the rsync directory have `0775` dir permission and a `0664` file permission set on them.

10. Move the branches to tags using the following command.

```
svn mv SRC-URL DEST-URL -m "Reason for this commit".
```

Notice

The original process in this document was voted on by the Geronimo community. Please formally propose all changes to dev@geronimo.apache.org.

See:

1. http://marc.theaimsgroup.com/?l=geronimo-dev&m=115094116905426&w=4

Revised process using maven tools voted on in March 2008. Only major structural changes now require votes.

See: (not yet in archive)