UsingNexus

Using Nexus for Commons Maven releases

DRAFT

Notes on using Nexus

NOTE - much of this content can be replaced simply by referencing the new unified docs: Publishing Maven Artifacts

What is Nexus?

Nexus is a repository manager, and acts as a staging repository which "intercepts" artifacts uploaded by **mvn deploy**.

Thus artifacts can be safely deployed to Nexus as part of voting on a release. The vote takes place on the staged artifacts. If the vote succeeds, the artifacts can be promoted to the live repository. If it fails, the artifacts can be deleted, and the process can restart.

It also allows redundant files (such as .asc.md5 and .asc.sha1 hashes) to be deleted before deployment.

Preparations for using Nexus

- All Commons components that use the *org.apache.commons* groupId are already set up to use Nexus.

However, components that use different groupIds may need to be set up. See the instructions in Publishing Maven Artifacts to request adjustments to the Nexus configuration to support additional groupIds.

- If you have not done so already, create a master password.

See http://maven.apache.org/guides/mini/guide-encryption.html The master password should be stored on a USB stick if your host system is not secure. Create the ~/.m2/settings-security.xml file, for example:

```
<settingsSecurity>
  <relocation>/USB/settings-security.xml</relocation>
</settingsSecurity>
```

This should point to the file containing the master password:

```
<settingsSecurity>
    <master>Master password goes here</master>
</settingsSecurity>
```

If you are sure that your system is secure, then the master password can be stored directly in the ~/.m2/settings-security.xml file.

- Ensure that you have set up your Apache username and password in your settings.xml file.

There are two sections that need to be defined:

- apache.snapshots.https
- apache.releases.https

These correspond with Nexus repository definitions in the Apache parent pom (v7).

```
<!-- To publish a snapshot -->
<server>
  <id>apache.snapshots.https</id>
  <username><!-- ASF login name --></username>
  <password>{encryptedpassword}</password>
</server>
<!-- To publish a release -->
<server>
  <id>apache.releases.https</id>
  <username><!-- ASF login name --></username>
  <password>{encryptedpassword}</password>
</server>
```

How to use deploy using Nexus

As noted above, there are two different repositories, one for snapshots and the other for releases. If the version contains the suffix **-SNAPSHOT**, then the snapshot repo will be chosen by **mvn deploy** otherwise deploy will use the release repo.

Creating a Nexus snapshot

In the run-up to a release, it is worth-while creating a snapshot. This will allow checking of the artifacts, as well as checking that the settings are correct.

To create the snapshot, make sure that the current pom.xml is for a SNAPSHOT version. Perform the following command:

```
mvn deploy [-DskipTests]
```

This will create the binary jar(s) and hashes and upload them.

If you want to check that the source and javadoc jars are correct, perform the following:

```
mvn deploy -Prelease -Dgpg.skip [-DskipTests]
```

This will create the binary, source and javadoc jars and hashes and upload them.

The `-Prelease` flag ensures that the source and javadoc jars are created. The `-Dgpg.skip` flag skips the signing phase; do not omit this as it might mislead users if a signed jar was deployed as a snapshot

The resulting snapshot will appear in the Nexus repository under:

https://repository.apache.org/content/repositories/snapshots/org/apache/commons/

If the upload fails with `Error deploying artifact: Failed to transfer file: ... Return code is: 401` then either the username /password are incorrect, or Nexus has not yet been set up for the commons component.

Deploying to a local directory

The deploy target can easily be overridden by defining the altDeploymentRepository property. For example, the following will deploy to the local directory `target/deploy`:

```
mvn deploy -Prelease [-Dgpg.skip] [-DskipTests] -DaltDeploymentRepository=id::default::file:target/deploy
```

It's difficult to type and remember this parameter. Therefore the Commons Parent POM (from version 16) contains the following profile:

```
<profile>
  <id>test-deploy</id>
  <properties>
    <altDeploymentRepository>id::default::file:target/deploy</altDeploymentRepository>
  </properties>
</profile>
```

So you can add `-Ptest-deploy` to the deploy command to change the deployment to use `target/deploy`.

For example:

```
mvn clean deploy -Prelease-Ptest-deploy
```

For Commons Parent, you can do:

```
mvn clean gpg:sign deploy -Ptest-deploy
```

Note: Per default, the `Built-by` header in the manifest of the generated jars is filled from your operating system's login ID. If your Apache ID differs from this ID, you can override the Maven default by specifying the `user.name` property at the command line, e.g.

```
mvn clean deploy -Prelease-Ptest-deploy
```

```
 -Duser.name=<apacheID>
```

Creating a Nexus staging release

Make sure you are using Commons Parent V16 or later.

This is necessary to ensure that Nexus is used as the deployment target.

Prepare Your Version Number

A guideline regarding version numbering can be found at http://commons.apache.org/releases/versioning.html - within Commons we reached the following consensus

- when your release a new major or minor version it comes without point release number, e.g "foo-1.0.jar" or "foo-2.1.jar"
- when doing a bugfix release your need to add the point release number, e.g "foo-1.0.1.jar" or "foo-2.1.1.jar"

Prepare Your Maven Variables

During the staging process a directory is created on *people.apache.org* based on the content of the following Maven variables

- commons.release.version is a duplicate of the pom version
- commons.rc.version is the current number of your release candidate

That could look like the following snippet taken from *commons-exec*

```
<properties>
  <commons.release.version>1.0.0</commons.release.version>
  <commons.rc.version>RC2</commons.rc.version>
</properties>
```

Update Your Download Page

The commons-build plugin generates the download page in `./src/site/xdoc/download_[commons-foo].xml` based on `commons.release.version`

```
mvn commons:download-page
```

Update the release notes

Ensure that the release notes refer to the release version, not snapshot. If using automatic release note generation from changes.xml, the following command should update the file:

```
mvn changes:changes-report changes:announcement-generate
```

The file generated is `target/announcement/announcement.vm`

Prepare Your Assembly Descriptors

If you are declaring/using your own assembly descriptors make sure that they are not using ${version} but ${commons.release.version} - there seems to be an odd bug which results in an incorrectly expanded version string, e.g. `commons-exec-2.4.1-src`. If in doubt look at `./src/assembly/src.xml`

Commit Your Changes

When you followed the instructions you have modified a couple of files by now - commit them now otherwise the release process will fail.

Create the SVN tags (Manual method)

Create a clean SVN workspace for the release candidate

`svn co https://svn.apache.org/repos/asf/commons/proper/<component>/trunk <component>-m.n.o-RC1`

Edit the version fields in the POMs to remove the -SNAPSHOT

For example, on Windows (requires perl):

Note: The snippet below assumes that you are running it as a bat file. If you are just typing these commands at the command prompt, replace "%%i" with "%i". For reference, see the help displayed by "help for" or the following Microsoft's online documentation.

```
cd <component>-m.n.o-RC1
FOR /F "usebackq delims==" %%i IN (`dir pom.xml/s/b`) DO ^
    perl -pi.bak -e "s!-SNAPSHOT</version!</version!" %%i
del /S pom.xml.bak
```

Or using Maven:

`mvn versions:set -DnewVersion=3.1 -DgenerateBackupPoms=false`

Edit the SCM entries in the parent POM. Note: use the final tag, without any RC suffix

Do "svn diff" and "svn st" to check that the changes are OK. These should only show changes to the POMs

Create the RC tag, by copying the tag workspace to SVN:

```
svn copy <component>-m.n.o-RC1 -m"Creating <component>-m.n.o-RC1 tag" https://svn.apache.org/repos/asf/commons/proper/<component>/tags/<component>-m.n.o-RC1
```

This should result in a new tag, containing the "trunk" code with the POM fixes only. All history is preserved.

Create the SVN tags (using Maven Release plugin)

Check that your poms will not lose content when they are rewritten during the release process.

- mvn release:prepare -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 backport 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.
- If everything is ok, run: mvn release:prepare

After this step the necessary SVN tags should have been created.

If you have problems running this command, the following profile in settings.xml may help:

```
<profile>
  <id>release</id>
  <properties>
     <gpg.keyname>your_key</gpg.keyname>
  </properties>
</profile>
```

Or running mvn with this arguments:

```
mvn -DdryRun=true -Dgpg.keyname=your_key \
-Darguments="-Dgpg.keyname=your_key" release:prepare
```

Deploy the artifacts

Once the code appears to be ready, the Maven artifacts can be deployed:

```
mvn deploy -Prelease [-Ptest-deploy]
```

The `-Ptest-deploy` parameter can be used as described above to redirect the deployment to target/deploy.

Otherwise, the artifacts will be deployed to the release staging repository in Nexus.

See Closing the Staged Repository for information about reviewing your staged artifacts and making them available for others to review.

Stage the site

The site plugin can be used for staging a website. You probably need to add some authentication information to your settings.xml. Please mind the id "stagingSite".

```
<server>
  <id>stagingSite</id>
  <username>repouser</username>
  <!-- other optional elements:
     <password>my_login_password</password>
     <privateKey>/path/to/identity</privateKey> (default is ~/.ssh/id_dsa)
     <passphrase>my_key_passphrase</passphrase>
  -->
</server>
```

Once done, run a command like this from your tag:

```
mvn site:stage-deploy -DstagingDirectory=src/site \
-DstagingSiteURL=scp://[...]/people.apache.org/builds/commons/compress/1.1/RC1
```

The site should appear on the specified folder. Alternatively, you can stage the site in your *public_html* folder in your home directory on *people.apache.org*. This can be done with the following command:

```
mvn site:stage-deploy
-DstagingSiteURL=scp://people.apache.org/home/<ApacheID>/public_html/foo-1.2rc1
```

Send Out The Vote

Below you find a vote template to save you some time ...

```
Tag:

https://svn.apache.org/repos/asf/commons/proper/YOUR_PROJECT/tags/${commons.rc.version}

Site:

http://people.apache.org/builds/commons/YOUR_PROJECT/${commons.release.version}/${commons.rc.version}/site
/index.html

Binaries:

Add the Nexus URL to your binary artifacts, f. e. https://repository.apache.org/content/repositories/snapshots
/org/apache/commons/commons-compress

[ ] +1 release it
[ ] +0 go ahead I don't care
[ ] -1 no, do not release it because
```

React to the Vote

If the vote failed, you need to Drop the repo and start again.

If the vote passed the following steps need to be performed:

Summarise the vote

Once the 72 hours have passed, send a **[VOTE][RESULT]** mail to summarise the outcome of the vote. This should tally the votes cast, and state which are binding (PMC members).

Note: if insufficient votes have been received by the end of 72 hours, this does not necessarily mean that the vote has failed. The vote can be held open longer if necessary, but should not normally be closed before the 72 hours have elapsed. (Exceptions may be made in special cases).

Releasing

NOTE The Commons Parent pom currently attaches the assembly archives (the -bin* and -src* files) to the deploy phase, so they end up being copied to the Nexus staging repository. This is convenient for voting, as all the files are in the same place, but means that additional action needs to be taken to move the non-Maven files to the correct place.

wget the non-Maven artifacts from Nexus to people server

In your home directory, perform:

```
wget -np -r https://repository.apache.org/content/repositories/orgapachecommons-098/org/apache/commons/commons-
foo/1.1/
```

Here is a specific example (used for commons-io 2.1 on 7 October 2011), where you must turn off robots.txt and accept a self-signed certificate:

```
wget -r -l 1 -np -nH -nd -nv -e robots=off --wait 10 --no-check-certificate https://repository.apache.org
/content/repositories/orgapachecommons-027/commons-io/commons-io/2.1/
```

For the curious:

- -r recursive
- -l 1 1 level
- -np no parent
- -nH don't create host directories
- -nd don't create directories
- -nv quiet
- -e robots=off ignore robots.txt
- --wait 10 wait between retrievals

Check the MD5/SHA hashes! This can be done for instance with the *verify_sigs.sh* script available under https://svn.apache.org/repos/private/committers/tools/releases (just check out from SVN and place the shell scripts in the *bin* folder below your home directory).

Also, change the group to commons and ensure that the files can be written by the group:

```
chgrp commons *
chmod g+w *
```

Copy to dist

On people.apache.org, change directory to the distribution directory for your component:

```
cd /www/www.apache.org/dist/commons/foo/
```

Move source distributions, their detached signatures and md5 sums into position. All source versions live in the source subdirectory.

```
mv ~/foo-1.2-RC3/commons-foo-1.2-src* source
```

Move the binary distributions, their detached signatures and md5 sums into position. All binary versions live in the binaries subdirectory.

```
mv ~/foo-1.2-RC3/commons-foo-1.2-bin* binaries
```

Double check the permissions for both binaries and source distributions. The file permissions should be "~~rw-rw-r-~~" and the group should be "commons", for example:

```
-rw-rw-r--  1 userid   commons     203 Feb 21 23:45 commons-foo-1.2-src.tar.gz.asc
```

Update README

The contents of the README.html are displayed at the bottom of the html showing the directory listing. This document should be updated to reflect the new release. If this document is not present, then copy one from an existing release directory and edit that. Update the latest release number. Please also read through and correct any mistakes you find and fix other items (eg. urls) which need updating. Copy the revised README.html into the binary and source directories, replacing any old versions.

Update RELEASE-NOTES

Replace the current RELEASE-NOTES.txt with the new release notes.

Symbolic Links

These are no longer to be used, so please do not use symbolic links in mirrored directories!

Nexus release

1. remove commons-compress-1.1-bin.* and commons-compress-1.1-src.* from nexus staging area
2. Promote

Note that Nexus release is asynchronous - it may take a short while for the files to be copied to the release area of Nexus and for the status to be updated.

It may take a while for the Maven repo to be synchronised from Nexus. Sometimes the files don't all appear at once.

Test

Wait until the release files are available from the main Apache web site (http://www.apache.org/dist/commons/foo/), then confirm things are good. Check the main directory:

1. Examine the directory listing page. At the bottom should be found the information you entered into the README.html. Please check that this is correct.
2. Check the KEYS file
3. Check the RELEASE-NOTES.txt
4. Download and verify the current distributions, the following might help committers/tools/releases/verify_sigs.sh.
5. Follow the links to the binaries and source directories. Check them in a similar manner.

Publish the website

See https://commons.apache.org/site-publish.html

Send announcement

Announce the availability of the new release. Please check that the mirrors have been updated with the new release before sending the announce. There is no need to check all mirrors, but it's worthwhile checking a few of them.

Please remember to give a brief description of your component. Please also remember to remind people about verifying the signatures. The subject should be something like [ANNOUNCEMENT] Foo 1.2 Released. Send this mail from your Apache account. Please spell check the document!

Wait to send the release announcement until you have verified that the release artifacts are available on the mirrors.

The component website including the updated download page has been updated on the public site http://commons.apache.org/foo.

If the component publishes maven artifacts, these artifacts have been replicated to the central maven repo at repo1.maven.org. (Clear your local repo of the release artifacts and either activate the clirr report with the updated version info or update a local project with a dependency to the new release version to get maven to try to download the release artifacts. Or just access repo1 using a web browser.) The release announcement should go to (at least) the following mailing lists:

- announce@apache.org
- dev@commons.apache.org
- user@commons.apache.org

Post Release Cleanup

The actual release task is now complete, however there are some remaining steps to do some cleanup and to prepare the next development iteration.

- **Remove Obsolete Releases**: Unless old versions are especially required, remove them from the dist directory. This will cause the files to also be deleted from the mirrors and save them some diskspace as well as simplifying things for users. Note that the contents of the /www/www.apache.org/dist directory is regularly copied to /www/archive.apache.org/dist and from there transferred to host archive.apache.org. Deleting files from the standard distribution directories does *not* delete them from the archive dist directories so users will still be able to access old files even when they are not available from the mirrors.
- **Update JIRA**: Mark the release as released in the JIRA project admin and CLOSE all issues RESOLVED in this release. Make sure the FIX VERSION for all issues closed with this release is set correctly to this version. If this has not already been done, create a JIRA version for the next release.
- **Update DOAP file**: Update the component's DOAP file with details of the released version:

```
<release>
  <Version>
    <name>x.y.z</name>
    <created>yyyy-mm-dd</created>
    <revision>x.y.z</revision>
  </Version>
</release>
```

- **Update changes.xml**: If the component uses the maven changes plugin to maintain its changelog, fill in the release date for the just released version and create a new release element for the next release.
- **Create the relase tag**: Copy the tag of the RC whose vote succeeded to the final release tag, e.g.

```
svn copy https://svn.apache.org/repos/asf/commons/proper/foo/tags/FOO_1.2RC1
  https://svn.apache.org/repos/asf/commons/proper/foo/tags/FOO_1.2 -m "Tagging 1.2 release"
```