Maven release plugin cheat sheet

⚠️ This page assumes that project poms are properly configured for performing a release and that the deployer’s settings.xml file is also properly configured.

**General rules**

1. Release from *.x branch.

**Prerequisites**

Maven 2.0.10 minimum, Maven 2.2.* recommended.

**Review merges/commits**

Confirm that all merges scheduled for release have been merged and committed to the target *.x branch.

**Check out project**

✅ Recommendation: check out a fresh copy of the project before performing a release. This ensures that no modified but uncommitted files are in your working copy. The release plugin will fail whenever it detects uncommitted files. Also avoid opening your fresh check out in Eclipse before performing a release; you will pollute the project with Eclipse uncommitted metadata files.

```bash
sakaiger$ svn co https://source.sakaiproject.org/svn/kernel/branches/kernel-1.1.x/ kernel-1.1.x
```

**Verify build**

✅ Recommendation: before performing release delete your local .m2/repository/org/sakaiproject/ProjectToRelease folder. Then install the project with the offline switch -o activated. If dependencies are missing from your local repo, the install will fail. Rerun the install without the -o switch to populate your repo. Then reinstall the project in offline mode again.

```bash
sakaiger$ rm -rf ~/.m2/repository/org/sakaiproject/kernel
sakaiger$ cd kernel-1.1.x
sakaiger$ mvn -o clean install
```

**Perform release dry run**

During the dry run and again during the release proper you will be prompted for version information. The release plugin will attempt to guess the release, tag and snapshot versions. You can accept the values as is or supply your own.

```bash
[INFO] Checking dependencies and plugins for snapshots ...  
What is the release version for "BasicLTI base pom (basiclti)"? (org.sakaiproject.basiclti:basiclti) 1.1.0-b02: :  
What is SCM release tag or label for "BasicLTI base pom (basiclti)"? (org.sakaiproject.basiclti:basiclti) basiclti-1.1.0-b02: :  
What is the new development version for "BasicLTI base pom (basiclti)"? (org.sakaiproject.basiclti:basiclti) 1.1.0-b03-SNAPSHOT: :
```

```bash
sakaiger$ cd kernel-1.1.x
sakaiger$ mvn release:prepare -DdryRun=true
```

**Perform release**

After a successful dry run, perform the release.

```bash
sakaiger$ mvn release:clean release:prepare release:perform
```

**Signing artifacts with GPG**
sakaiger$ mvn -Pgpg release:clean release:prepare release:perform

OAE 3akai-ux release
sakaiger$ mvn -Psakai-release,gpg release:clean release:prepare release:perform

release:prepare goal actions:
- Prompt user for release version info (name, number, next x-SNAPSHOT)
- Check that there are no uncommitted changes in the source
- Check that there are no SNAPSHOT dependencies
- Change the version in the POMs from x-SNAPSHOT to a new version
- Transform the SCM information in the POM to include the final destination of the tag
- Run the project tests against the modified POMs to confirm everything is in working order
- Commit the modified POMs
- Tag the code in the SCM with a version name
- Bump the version in the POMs to a new value y-SNAPSHOT
- Commit the modified POMs

release:perform goal actions:
- Checks out release tag based on <scm> URL
- Builds release tag
- Deploys release binaries to remote repository (e.g. runs pre-defined Maven release goals: deploy and site-deploy)

release:clean goal actions:
- Removes release.properties and other release-related files from checkout.

Perform release without generating a site
If a <site> is defined in <DistributionManagement> you can prevent site generation by adding the optional goals parameter -Dgoals=deploy.

Using the goals parameter you can specify a list of goals to execute on deployment. Default value is either deploy or deploy site-deploy if a <site> is defined.
sakaiger$ mvn release:clean release:prepare release:perform -Dgoals=deploy

Using Profiles
You can activate profiles when running the release plugin. Test & Quizzes (Samigo) requires a signed audio jar. This is accomplished by adding a profile in one's local settings.xml file that provides the Keystore location, password and alias.

Profile activation
sakaiger$ mvn -Pjarsign release:clean release:prepare release:perform

Sitestats Impl includes performance tests. If the profile is not activated the following build error will be generated:

```
[INFO] ----------------------------------------
[ERROR] FATAL ERROR
[INFO] ----------------------------------------
[INFO] includes1 has null value
[INFO] ----------------------------------------
[INFO] Trace
java.lang.NullPointerException: includes1 has null value
    ...
```

Profile activation
sakaiger$ mvn -Pperftests release:clean release:prepare release:perform

settings.xml

<profiles>
  <profile>
    <id>jarsign</id>
    <activation>
      <activeByDefault>false</activeByDefault>
    </activation>
    <properties>
      <keystore.location>KEYSTORE-LOCATION</keystore.location>
      <keystore.alias>ALIAS</keystore.alias>
      <keystore.password>PASSWORD</keystore.password>
    </properties>
  </profile>
  ...<activeProfiles>
  <activeProfile>jarsign</activeProfile>
</activeProfiles>

Release Gotchas

Repo permission issues

If you encounter a "Permission denied" error when executing the release:perform goal, you do NOT need to rollback the release. Instead, address the permission issue and then check out your newly minted tag and perform the release using the tag itself.

Note: If you get this error about permission denied

- Check your ~/.m2/settings.xml and verify that you have a server block which matches up the appropriate distributionManagement blocks. Otherwise it will attempt to login with the wrong user (the user you're logged in as). You may also need to specify a <privateKey> </privateKey> location, if it's not your default private key. See this for more info: http://maven.apache.org/plugins/maven-deploy-plugin/examples/deploy-ssh-external.html

<servers>
      <id>sakai-maven2-scp</id>
      <username>shared_release</username>
</servers>

sakaiger$ svn co https://source.sakaiproject.org/svn/jobscheduler/tags/scheduler-2.7.0-rc06
sakaiger$ cd scheduler-2.7.0-rc06
sakaiger$ mvn release:perform -DconnectionUrl=scm:svn:https://source.sakaiproject.org/svn/jobscheduler/tags/scheduler-2.7.0-rc06

Release rollback

If the release should fail you can rollback the release as follows:

mvn release:rollback

Rolling back a release

release rollback goal actions:
All project POMs are reverted back to their pre-release state locally, and also in the SCM if the previous release command was able to successfully make changes in the SCM to the POMs. This is done by using the backup files created during release:

- If a tag was generated successfully before a plugin failure it will have to be removed manually.
- Release rollbacks will fail if `mvn release:clean` is executed prior to the rollback attempt.

Should the build fail rerun the following `mvn` goals to restart the build:

```
sakaiger$ mvn clean install
sakaiger$ mvn release:prepare
```

**Release from a tag**

If a release fails after a tag has been generated and the branch pom versions have been incremented there should be no need to perform a `mvn release:rollback`. Instead check out the tag and perform the release using it.

```
mvn -Pjarsign release:perform -DconnectionUrl=scm:svn:https://source.sakaiproject.org/svn/sam/tags/samigo-2.7.1
```

**Spurious snapshot resolution issues**

Code that you plan to release should never include dependencies on unstable snapshot code. Obligingly, the release plugin will warn you if it encounters snapshot dependencies during the `release:prepare` phase. Occasionally, however, the plugin will mishandle version ranges specified in a dependency (e.g., common [1.0.0,1.1.0]) and will issue spurious warnings that snapshots require resolving. This occurs whenever the plugin finds snapshot versions of project dependencies resident in your local `.m2` repo. The workaround is to delete the `.m2` project artifacts folder (e.g., `.m2/repository/org/sakaiproject/common`) and then issue a `mvn clean install` to install the required dependency. Note, that you may have to delete more than one folder from your `.m2` repo in order to eliminate the plugin’s dependency resolution challenges.

```
[INFO] Building Profile2
[INFO]   task-segment: [release:prepare] (aggregator-style)
[INFO]
[INFO] [release:prepare {execution: default-cli}]
[INFO] Resuming release from phase 'check-dependency-snapshots'
[INFO] Checking dependencies and plugins for snapshots ...
There are still some remaining snapshot dependencies.: Do you want to resolve them now? (yes/no) no: : yes
Dependency type to resolve,: specify the selection number ( 0:All 1:Project Dependencies 2:Plugins 3:Reports 4:Extensions ): (0/1/2/3) 1: : 0
Resolve All Snapshots.: 'org.sakaiproject.common:sakai-common-edu-person-api' set to release? (yes/no) yes: : yes
[INFO]
```

```
[ERROR] FATAL ERROR
[INFO]
```

```
[INFO] null
[INFO]
```

```
[INFO] Trace
java.lang.NullPointerException ...
```

**Spurious svn commit error ("File ... already exists")**

Encountered the following build failure during `release:prepare` when releasing remotely from UCT’s arabica.cet.uct.ac.za.
[ERROR] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Unable to tag SCM
Provider message:
The svn tag command failed.
Command output:
svn: Commit failed (details follow):
svn: File '/svn/pure-poms/tags/purepoms-2.7.2/sakai-basic-tool/pom.xml' already exists

A quick check of the svn repo confirmed that no purepoms-2.7.2 tag existed. Even an svn delete confirmed that the tag was a phantom.

sakaiger$ svn del https://source.sakaiproject.org/svn/pure-poms/tags/purepoms-2.7.2 -m "NOJIRA delete phantom tag"
svn: URL 'purepoms-2.7.2' does not exist

The issue can be resolved by issuing an svn update immediately after encountering the release plugin build failure followed by release:prepare (repeat) and then release:perform.

[ERROR] BUILD FAILURE ...
sakaiger$ svn update
sakaiger$ mvn release:prepare release:perform

For more info see: MRELEASE-409.

No SNAPSHOT project in the reactor projects list

Encountered the following build failure during release:prepare when releasing SiteStats 2.1.1.

[INFO] ------------------------------------------------------------------------
[ERROR] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] You don't have a SNAPSHOT project in the reactor projects list.
[INFO] For more information, run Maven with the -e switch
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 17 seconds
[INFO] Final Memory: 5M/11M
[INFO] ------------------------------------------------------------------------

This issue is caused when the version on the tool pom.xml (and all tool modules pom.xml files) does not end with the `-SNAPSHOT` suffix. In this specific case, tool version was '2.1.1-b01' instead of '2.1.1-b01-SNAPSHOT'.

To fix this, just need to add the suffix `-SNAPSHOT` to the version tag of tool pom.xml and tool modules pom.xml files, commit the changes, and try again.

maven-javadoc-plugin JVM settings truncated or not all included on a single line.

If you accidently format the pom XML in such a way that the JVM settings for the maven-javadoc-plugin are divided across two lines you will encounter the following error:

```xml
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-javadoc-plugin</artifactId>
<reportSets>
  <reportSet>
    . .
    <additionalJOption>
      -J-Xms168m -J-Xmx512m
      -J-XX:PermSize=128m -J-XX:NewSize=64m
    </additionalJOption>
  </additionalJOption>
```
Release plugin fails during site generation. This issue is related to https://jira.codehaus.org/browse/SUREFIRE-714.
maven-surefire-plugin and maven-surefire-report-plugin 2.7.1+ requires maven-site-plugin 2.1+.

Release fails when attempting to generate Javadocs due to packages having commented out/missing source files
See https://jira.sakaiproject.org/browse/SAK-20961

In this instance the release plugin failed on common when generating Javadocs due to the presence of two commented out test classes.
[INFO] Loading source files for package org.sakaiproject.api.common.type...
[INFO] Loading source files for package org.sakaiproject.api.common.edu.person...
[INFO] Constructing Javadoc information...
[INFO] 1 error
[INFO] 2 warnings
[INFO] [INFO] ------------------------------------------------------------------------
[INFO] [ERROR] BUILD ERROR
[INFO] [INFO] ------------------------------------------------------------------------
[INFO] [INFO] Error during page generation
[INFO]
[INFO] Embedded error: Error rendering Maven report:
[INFO] [INFO] Exit code: 1 - javadoc: warning - No source files for package org.sakaiproject.api.common.type
[INFO] javadoc: warning - No source files for package org.sakaiproject.api.common.edu.person
[INFO] javadoc: error - No public or protected classes found to document.
[INFO]
[INFO] Command line was: /usr/lib/jvm/java-6-sun-1.6.0.24/jre/../bin/javadoc -J-Xmx1024m -J-Xms168m "-J-Xms168m -J-Xmx1024m -J-XX:PermSize=128m -J-XX:NewSize=64m" @options @packages
[INFO]
[INFO] Refer to the generated Javadoc files in '/home/arwhyte/svn/common/branches/common-1.0.x/target/checkout/target/site/testapidocs' dir.

The issue was fixed by defining an `<excludePackageNames>` setting to the Javadoc plugin `<configuration>`:

```xml
<!-- SAK-20961 exclude commented out test classes -->
<excludePackageNames>org.sakaiproject.api.common.edu.person:org.sakaiproject.api.common.type</excludePackageNames>
</plugin>
```

**SNAPSHOT deployments**

If you need to deploy or refresh a set of snapshot binaries in our maven2 snapshot repo issue the following command:

```bash
sakaiger$ cd msgcntr-trunk
sakaiger$ mvn clean install javadoc:jar source:jar deploy
```

Profile activation

```bash
sakaiger$ cd samigo-trunk
sakaiger$ mvn -Pjarsign clean install javadoc:jar source:jar deploy
```

**References**

- Maven Release Plugin
- Maven mini-guide
- J. Ferguson, "Using the Maven Release Plugin"