How To Release

This page is prepared for Apache Avro committers. You need committer rights to create a new Apache Avro release.

Branching

Skip this section if this is NOT the first release in a series (i.e. release X.Y.0).

1. Update `CHANGES.txt` to include the release version and date (use `Unreleased` for the date if it is unknown) and add a new section, `Master (unreleased changes)`.
2. Commit these changes to trunk.

   ```
   git commit -am "Preparing for release X.Y.Z"
   ```

3. Create a release branch for the release series:

   ```
   git checkout -b branch-X.Y
   ```

4. Update `CHANGES.txt` remove the `Master (unreleased changes)` section because it isn't used in the release branch.
5. Commit the update to the release branch.

   ```
   git commit -am "Preparing for release X.Y.Z"
   ```

6. Return to the master branch.

   ```
   git checkout master
   ```

7. Update the default version in `share/VERSION.txt` on trunk to X.(Y+1).0-SNAPSHOT. Be sure not to leave a trailing newline.
8. Update the version in the Maven POM files to match with

   ```
   mvn versions:set -DnewVersion=X.(Y+1).0-SNAPSHOT
   -DgenerateBackupPoms=false
   ```

9. Commit these changes to the branch.

   ```
   git commit -am "Preparing for X.(Y+1).0 development"
   ```

10. Push the master and branch changes:

    ```
    git push apache master
    git push apache branch-X.Y
    ```

Updating Release Branch

These operations take place in the release branch.

1. Check out the branch with:

   ```
   git checkout branch-X.Y
   ```

2. Update `CHANGES.txt` to include the release version and date (this change must be committed to trunk and any intermediate branches between trunk and the branch being released).
3. Update the version number in `share/VERSION.txt` to be "avro-X.Y.Z-SNAPSHOT". Be sure not to leave a trailing newline.
4. Update the version in the Maven POM files to match with

   ```
   mvn versions:set -DnewVersion=X.Y.Z-SNAPSHOT
   -DgenerateBackupPoms=false
   ```

5. Commit these changes.

   ```
   git commit -am "Preparing for release X.Y.Z"
   ```

6. Add the fix version X.Y.Z to the Avro JIRA
7. If not already done, cherry-pick desired patches from trunk into the branch and commit these changes. You can find the revision numbers using `svn log CHANGES.txt` in the branch and in trunk.

   ```
   git checkout branch-X.Y
   git cherry-pick <commit>
   ```

8. For each patch merged, change the fix version for the JIRA issue to be X.Y.Z
9. Go through CHANGES.txt, JIRA, and svn log to be sure that the issues included in the branch match in each location, then update the date in CHANGES.txt to be today.
10. Update the version number in `share/VERSION.txt` to be "X.Y.Z". Be sure not to leave a trailing newline.
11. Update the version in the Maven POM files to match with

    ```
    mvn versions:set -DnewVersion=X.Y.Z -DgenerateBackupPoms=false
    ```

12. Update the version number in `lang/c/version.sh` (the variables `libavro_micro_version`, `libavro_interface_age` and `libavro_binary_age`) according to the libtool versioning rules as described in that file. Note the libtool version number is completely unrelated to the Avro release version number.
13. Commit these changes.

    ```
    git commit -am "Preparing to build X.Y.Z"
    ```

14. Tag the release candidate (R is the release candidate number):

    ```
    git tag -s release-X.Y.Z-rcR -m "Avro X.Y.Z rcR release."
    ```

15. Push the release tag and branch changes:

```
git push apache release-X.Y.Z-rcR
git push apache branch-X.Y
```

Building

Unless you have set up the required dependencies to build Avro for all languages, the following should be run inside a Docker container (see *BUILD.txt* for instructions).

1. Build the release & run unit tests.

   ```
   ./build.sh clean dist test
   ```

2. Check that release files look ok - e.g. unpack the sources and run tests.
3. Sign the release (see Step-By-Step Guide to Mirroring Releases for more information).

   ```
   ./build.sh sign
   ```

   To sign a release, your key must be present in the dist/KEYS file. See the Apache guide to Signing Releases for more details. Also, if you've updated the dist/KEYS file, be sure to update the public Apache KEYS file as well:

   ```
   ssh people.apache.org
   cd /www/www.apache.org/dist/avro
   svn up
   ```

4. Copy release files to the public staging area https://dist.apache.org/repos/dist/dev/avro/

   ```
   svn co https://dist.apache.org/repos/dist/dev/avro/ avro-dev-dist
   mkdir avro-dev-dist/avro-X.Y.Z-rcR
   cp -pr avro/dist/* avro-dev-dist/avro-X.Y.Z-rcR
   cd avro-dev-dist
   svn add avro-X.Y.Z-rcR
   svn commit -m "Artifacts for Avro X.Y.Z RCR"
   ```

5. Before `1.9.x` stage the *Hadoop 1* version of the Java artifacts to the Maven repository:

   ```
   mvn clean -P dist,sign deploy -DskipTests=true -Dhadoop.version=1
   -Dgpg.passphrase=XXX -Davro.version=X.Y.Z
   ```

   (From `1.9.x` we do not support *Hadoop 1* anymore. All the Java artifacts are created for Hadoop 2 by default.)
6. Stage the default *Hadoop 2* version of Java artifacts to the Maven repository:

   ```
   mvn clean -P dist,sign deploy -DskipTests=true -Dgpg.passphrase=XXX
   -Davro.version=X.Y.Z
   ```

7. Find the Staging Repository, and close it.

8. Call a release vote on dev at avro.apache.org.
   Include the URL of the staging repository.

Publishing

Once three PMC members have voted for a release, it may be published.

1. Tag the release:

```
git checkout release-X.Y.Z-rcR
git tag -s release-X.Y.Z -m "Avro X.Y.Z release."
```

2. Copy release files to the release repository.

```
svn copy https://dist.apache.org/repos/dist/dev/avro/avro-X.Y.Z-rcR \
https://dist.apache.org/repos/dist/release/avro/avro-X.Y.Z -m "Avro
X.Y.Z release."
```

3. The release directory usually contains just two releases, the most recent from two branches, with a link named 'stable' to the most recent recommended version.

```
svn co https://dist.apache.org/repos/dist/release/avro/
avro-release-dist
cd avro-release-dist
svn rm avro-A.B.C; rm stable
ln -s avro-X.Y.Z stable
svn commit -m  "Avro X.Y.Z release."
```

4. Publish Java artifacts to the Maven repository:
   Find the Staging Repository and release it.
5. Publish Python artifacts to PyPI. To do this you'll need an account on PyPi, and write access to the Avro package - ask the existing owners for permission if you don't have it.

```
mkdir -p tmp/py
cd tmp/py
tar xzf ../../dist/py/avro-X.Y.X.tar.gz
cd avro-X.Y.Z
python setup.py sdist upload
```

6. Publish Python3 artifacts to PyPI.

```
mkdir -p tmp/py3
cd tmp/py3
tar xvf ../../dist/py3/avro-python3-X.Y.Z.tar.gz
cd avro-python3-X-Y-Z
python3 ./setup.py sdist upload
```

7. Publish Ruby artifacts to RubyGems. Again, you'll need an account and you need to be an owner.

```
gem push dist/ruby/avro-X.Y.Z.gem
```

8. Publish JavaScript artifacts to NPM. Again, you'll need an account and you need to be an owner.

```
npm publish dist/js/avro-js-X.Y.Z.tgz
```

9. Wait 24 hours for release to propagate to mirrors.
10. Prepare to edit the website.

```
svn co https://svn.apache.org/repos/asf/avro/site
```

11. Update the documentation links in `author/content/xdocs/site.xml`.
12. Update the release news in `author/content/xdocs/releases.xml`.
13. Regenerate the site, review it, then commit it. Note that Forrest 0.9 is easiest (not currently in the Docker image), since it doesn't require Java 5.

```
cd site
ant
firefox publish/index.html
svn commit -m "Updated site for release X.Y.Z."
```

14. Copy the new release docs to website and update the `docs/current` link:

```
tar xzf dist/avro-doc-X.Y.Z.tar.gz
mv avro-doc-X.Y.Z ../site/publish/docs/X.Y.Z
cd ../site/publish
svn add X.Y.Z
rm current
ln -s X.Y.Z current
svn commit -m "Adding documentation for release X.Y.Z."
```

15. Send announcements to the user and developer lists once the site changes are visible.
16. Update the version number in `share/VERSION.txt` to be "avro-X.Y.N-SNAPSHOT", where "N" is one greater than the release just made.
17. Update the version in the Maven POM files to match with

```
mvn versions:set -DnewVersion=X.Y.N-SNAPSHOT
-DgenerateBackupPoms=false
```

18. In Jira, ensure that only issues in the "Fixed" state have a "Fix Version" set to release X.Y.Z.
19. In Jira, "release" the version. Visit the "Administer Project" page, then the "Manage versions" page. You need to have the "Admin" role in Avro's Jira for this step and the next.
20. In Jira, close issues resolved in the release. Disable mail notifications for this bulk change.

See Also

- Apache Releases FAQ

- Apache Infrastructure
- Apache Site Publishing Details