Release Process

- Prerequisites
  - Discuss
  - Prepare
    - Start with clean local repo
    - Switch to the branch that you will branch from
  - Tag Branch Point and Branch
  - Switch to the new branch
  - Clone & Checkout Branch
    - Cherry pick previous commits from master
    - Search for and replace all occurrences of the previous branch versions within the project's files
    - Update version numbers on master branch (from A.B.C-SNAPSHOT to D.E.F-SNAPSHOT) and push changes
    - Update version numbers on release branch (from A.B.C-SNAPSHOT to A.B.C) and push changes
    - Update CHANGES
    - Update documentation
    - Build, Test and Push Changes
    - Create a new Jenkins job to build the release. It should call these two commands. It is probably best to copy the previous releases job.
      - Download the release candidate
  - Sanity Test
- Sign
  - Verify Signatures
  - Tag Release Candidate
  - Stage
  - Community reviews the RC
- Vote
  - Iterate based on feedback until vote passes
  - Once vote passes, tag the release:
    - Promote
    - Verify that the results are accessible.
- Publish to Maven Repository
  - Preparation
  - Release
  - Wait 24 hours for release to propagate to mirrors.
  - Update site
  - Create version in JIRA for release X.Y.
  - Send announcements to the user and developer lists.
  - Update CHANGES with header for new changes
  - Remove old release from dist.apache.org
  - Trademarks
  - License
  - Privacy Policy

Prerequisites

Before entering into this process you need to ensure you will be able to cryptographically sign the final result in such a way that others can validate the signature. This can be a confusing process. Here are links to several documents that should help.

- http://httpd.apache.org/dev/verification.html

Discuss

Send a [DISCUSS] email to the dev@knox list proposing a release.

Prepare

In preparation for each release there are a number of sub-steps required to ensure that the project’s repository is in a suitable state for branching.

Start with clean local repo

This can be a fresh clone or just a repo that has no pending changes or extraneous files in.
Switch to the branch that you will branch from

In this step we need to check out the previous release branch or some other existing branch from which we will create our new one.

List existing branches:

```
switch to the branch that you will branch from

list existing branches:

  git branch -al

  * master
    remotes/origin/HEAD -> origin/master
    remotes/origin/master
    remotes/origin/v0.2.0
    remotes/origin/v0.3.0
    remotes/origin/v{X.Y.Z-1}

check out the previous (release) branch:

  git checkout -t origin/v{X.Y.Z-1}

Tag Branch Point and Branch

This step tags the point in time of when the branching starts within the remote repo.

```

tag --annotate v{X.Y.Z}-branch --message "Branch point for v{X.Y.Z}"

tag push origin --tags
```

Switch to the new branch

Check out the branch that is intended for this release:

```
switch to the new branch

check out the branch that is intended for this release:

  git checkout -b v{X.Y.Z}
  git push --set-upstream origin v{X.Y.Z}
```

Clone & Checkout Branch

This step does a couple things for us:

- it ensures that the branch is actually there and available
- it ensures that our local repository to work from is clean

```
clone & checkout branch

this step does a couple things for us:

  git clone -b v{X.Y.Z} https://gitbox.apache.org/repos/asf/knox.git knox-{X.Y.Z}
cd knox-{X.Y.Z}
```

Cherry pick previous commits from master

You can use cherry-pick to pull commits in from existing branches.

To pull the changeset for the commit at the tip of the master branch:

```
cherry pick previous commits from master

you can use cherry-pick to pull commits in from existing branches.

to pull the changeset for the commit at the tip of the master branch:

  git cherry-pick origin/master
```

Search for and replace all occurrences of the previous branch versions within the project's files

All build artifacts that contain the previous branch's version need to be updated with the new version to reflect this new branch.
grep -r "0.3.0" .

Change all occurrences as appropriate.

Update version numbers on master branch (from A.B.C-SNAPSHOT to D.E.F-SNAPSHOT) and push changes

Update version numbers on release branch (from A.B.C-SNAPSHOT to A.B.C) and push changes

Update CHANGES

Update documentation

Build, Test and Push Changes

```
git pull
```
```
git commit --all --message "Updating branch."
```
```
ant verify
```
```
git push
```

Create a new Jenkins job to build the release. It should call these two commands. It is probably best to copy the previous releases job.

**New Jenkins Job**

Assuming that you have proper karma for creating new Jenkins jobs, you will see a link to create a new one. From that link you will be provided a page to select how to proceed; select copy existing job.

The Copy from text box will auto complete as you type - start with "Knox-" and select the job to copy from.

Ensure that the following form reflects the following values within various form elements:

- maven
- -Ppackage,release clean install
- ant post-build

Be sure to change any versions to reflect "v{X.Y.Z}"

Upon successful creation of the new job, you may manually kick off a build with the Build Now button.

**Download the release candidate**

You will be prompted for your Jenkins username and password.

```
ant download-candidate
```

**Sanity Test**

Do some basic manual testing to see if release looks ok. For example do and install and run through a few of the samples.

**Sign**

You will be prompted for your GPG passphrase.

```
ant sign-candidate
```

This will prompt you for your passphrase for each signed archive.
### Verify Signatures

Verify the hashes and signatures. First change into the distribution directory.

```bash
cd candidate
```

Verify the signatures for both the source and binary distribution. *Note: This assumes that gpg is installed.*

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>export KNOX_VERSION={X.Y.Z}</code></td>
</tr>
<tr>
<td><code>gpg --verify knox-${KNOX_VERSION}-src.zip.asc knox-${KNOX_VERSION}-src.zip</code></td>
</tr>
<tr>
<td><code>gpg --verify knox-${KNOX_VERSION}.zip.asc knox-${KNOX_VERSION}.zip</code></td>
</tr>
<tr>
<td><code>gpg --verify knoxshell-${KNOX_VERSION}.tar.gz.asc knoxshell-${KNOX_VERSION}.tar.gz</code></td>
</tr>
</tbody>
</table>

Verify the SHA-256 and SHA-512 hashes for both the source and binary distribution. *Note: This assumes a Linux or MacOS environment with openssl installed.*

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>export KNOX_VERSION={X.Y.Z}</code></td>
</tr>
<tr>
<td><code>cat knox-${KNOX_VERSION}-src.zip.sha256 &amp;&amp; openssl sha256 knox-${KNOX_VERSION}-src.zip</code></td>
</tr>
<tr>
<td><code>cat knox-${KNOX_VERSION}-src.zip.sha512 &amp;&amp; openssl sha512 knox-${KNOX_VERSION}-src.zip</code></td>
</tr>
<tr>
<td><code>cat knox-${KNOX_VERSION}.zip.sha256 &amp;&amp; openssl sha256 knox-${KNOX_VERSION}.zip</code></td>
</tr>
<tr>
<td><code>cat knox-${KNOX_VERSION}.zip.sha512 &amp;&amp; openssl sha512 knox-${KNOX_VERSION}.zip</code></td>
</tr>
<tr>
<td><code>cat knoxshell-${KNOX_VERSION}.zip.sha256 &amp;&amp; openssl sha256 knoxshell-${KNOX_VERSION}.zip</code></td>
</tr>
<tr>
<td><code>cat knoxshell-${KNOX_VERSION}.zip.sha512 &amp;&amp; openssl sha512 knoxshell-${KNOX_VERSION}.zip</code></td>
</tr>
</tbody>
</table>

### Tag Release Candidate

```
git tag --annotate vX.Y.Z-rcN --message "vX.Y.Z-rcN"
git push origin --tags
```

### Stage

Follow the instructions output by the sign step above. Basically execute this command. You will be prompted for your SVN username and password.

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cd ..</code></td>
</tr>
<tr>
<td><code>ant stage-candidate</code></td>
</tr>
</tbody>
</table>

### Community reviews the RC

[https://dist.apache.org/repos/dist/dev/knox/](https://dist.apache.org/repos/dist/dev/knox/)

You will be prompted for your SVN username and password.

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ant download-stage verify-stage</code></td>
</tr>
</tbody>
</table>

### Vote

Send a [VOTE] email to the dev@knox list. A template was output by the sign step above as `target/vote.txt`.

Iterate based on feedback until vote passes
Once vote passes, tag the release:

```
git tag --annotate v(X.Y.Z)-release --message "Release of v(X.Y.Z)"
git push origin --tags
```

**Promote**

You will be prompted for your SVN username and password.

```
ant promote-release
```

**Verify that the results are accessible.**


**Publish to Maven Repository**

**Preparation**

- Setup your `~/.m2/settings.xml` file as described [here](#).
- Make sure you encrypt your passwords as described [here](#).

**Staging**

This special variant of the build command will build and publish the release to a staging area in the Apache Nexus repo. Note: Get your gpg passphrase in your paste buffer you will need it MANY times. If someone can figure out how to use gpg-agent properly they should document it.

This page contains some information about getting gpg-agent installed and running.

```
mvn -Papache-release -Drepo.id=apache.releases.https deploy
```

If you have issues with the above command due to javadoc warnings, something like this can be done:

```
mvn -Dmaven.javadoc.failOnError=false -Papache-release -Drepo.id=apache.releases.https deploy
```

**Release**

Once that completes, login to the Apache Maven Nexus staging repositories with your Apache credentials and:

1. Select "Staging Repositories" from "Build Promotion" on the left.
2. Close the stage with the Close button at the top of the repo list. For the Description field use Apache Knox 0.5.1 Staged
3. Release the stage with the Release button at the top of the repo list. For the Description field Apache Knox 0.5.1 Release

**Wait 24 hours for release to propagate to mirrors.**

**Update site**

Update news in News.

**Create version in JIRA for release X.Y.**

Send announcements to the user and developer lists.

**Update CHANGES with header for new changes**
Remove old release from dist.apache.org

```
svn delete https://dist.apache.org/repos/dist/release/knox/${OLD_VERSION} -m "Delete Knox ${OLD_VERSION} from
dist.apache.org"
```

**Trademarks**

Apache Knox Gateway, Apache, the Apache feather logo and the Apache Knox Gateway project logos are trademarks of The Apache Software Foundation. All other marks mentioned may be trademarks or registered trademarks of their respective owners.

**License**

Apache Knox uses the standard [Apache license](#).

**Privacy Policy**

Apache Knox uses the standard [Apache privacy policy](#).