Release Engineering

N.B. Work in progress!

Release variants overview

Alpha, Beta and snapshot releases are made with git flow release start but are never finished, this is to change version and make tar-balls but we do not merge those changes back into develop.

Release and release candidate are connected, these are made with full git flow release process but with a few important difference.

- The git flow release is started with the release version (not as release candidate) and published.
- The version and NEWS is changed to the release candidate version.
- A tar-ball is created for the release candidate and upon success of that a tag is manually added.
- The release enters the one week candidate period and if blocking issues are found the release is aborted.
- The version and NEWS is changed to the release version.
- A tar-ball is created for the release and upon success the release is finished with git flow release finish and merged back into develop.

Preparation of release management host

Release tar-balls should be built AS A NORMAL USER on Ubuntu 12.04 (Server or Client does not matter) and these packages should be installed:

```
apt-get install build-essential libsqlite3-dev sqlite3 libbotan1.10-dev libssl-dev autoconf automake libtool
libcppunit-dev subversion git
```

Botan needs to be manually compiled to insure all functionality for SoftHSMv2, use the latest 1.10.x release.

```
cd /usr/src
wget http://botan.randombit.net/files/Botan-1.10.x.tgz
tar zxvf Botan-1.10.x.tgz
cd Botan-1.10.x.tgz

# You can cut&paste this:
./configure.py &&
make &&
make install &&
echo "OK" || echo "FAILED"
```

Run ldconfig to refresh the libraries.

```
ldconfig
```

How to make a release with a release candidate

1. Make sure it builds fine and all tests passes on Jenkins.
2. Create the release branch and publish it.
   New version should only be the version number, should not include the release candidate tag and should not include the software (ex for OpenDNSSEC 1.3.4 it would be just 1.3.4).

   ```
   # Make sure we have the latest updates
   git fetch --all -p
   # Delete local branches are check out them again
   git checkout develop
   git branch -D master
   git checkout master
   git branch -D develop
   git checkout develop
   # Make the release branch
   git flow release start <NEW VERSION>
   # Publish release branch
   git flow release publish <NEW VERSION>
   ```

3. Update version and NEWS with the <NEW VERSION> and the release candidate tag (rc<release candidate number>).

```
vi configure.ac
vi NEWS
vi ...
git commit -a -m "Update version and NEWS for release <NEW VERSION>rc<release candidate number>"
git push
```

4. Check out the release on release management host:

```
git clone https://github.com/opendnssec/<SOFTWARE>.git
cd <SOFTWARE>
git checkout master
git checkout develop
git flow init -d
git flow release track <NEW VERSION>
```

5. Run prepdist.sh to make a tar-ball:

```
# You can cut&paste this:
source prepdist.sh &&
make dist &&
echo "OK" || { echo "FAILED"; cd ..; }
```

6. Extract the tar-ball and make a full build, make sure that the tar-ball and the directory is correctly named.
   If "make check" does not exist, skip it and make the steps manually.

```
tar zxvf <SOFTWARE>-<NEW VERSION>rc<release candidate number>.tar.gz
cd <SOFTWARE>-<NEW VERSION>rc<release candidate number>

# You can cut&paste this:
source ../../prepdist.sh &&
make &&
make check &&
make install &&
rm -rf /tmp/*-release &&
echo "OK" || { echo "FAILED"; cd ..; }
```

7. Retrieve the release candidate tar-ball from the Ubuntu server and put it in distribution repository under source/testing.
8. Sign and checksum the tar-ball using the script sign-distfile.sh (available from the non-public OpenDNSSEC repository).

```
cd <corporation repository>/development/pgp
sh sign-distfile.sh <distribution repository>/source/testing/<SOFTWARE>-<NEW VERSION>rc<release
candidate number>.tar.gz
```

9. Verify the signature, first add the public key if you already haven't.

```
cd <corporation repository>/development/pgp
gpg --import opendnssec-pubring.gpg
```

   Now verify the signed release.

```
cd <distribution repository>/source/testing
gpg --verify <SOFTWARE>-<NEW VERSION>rc<release candidate number>.tar.gz.sig <SOFTWARE>-<NEW
VERSION>rc<release candidate number>.tar.gz
```

10. Commit all distribution files (tar-ball, detached signature and checksum files) to the distribution repository.

```
cd <distribution repository>/source/testing
svn add <SOFTWARE>-<NEW VERSION>rc<release candidate number>.tar.gz*
svn ci -m "<SOFTWARE> <NEW VERSION>rc<release candidate number>" <SOFTWARE>-<NEW VERSION>rc<release
candidate number>.tar.gz*
```

11. Tag the release candidate.

```
git tag -m "Tagging <SOFTWARE> <NEW VERSION>rc<release candidate number> release" <NEW
VERSION>rc<release candidate number>
git push --tags
```

12. Announce the release candidate, this is often done by the project leader but might be done by yourself if needed.
    Wait for the release to exit the release candidate period.
    If no blocking issues are found continue this process.
13. Update version and NEWS with the <NEW VERSION> (without the release candidate number).

```
# Update version for SoftHSM
vi configure.ac
vi NEWS
vi ...
git commit -a -m "Update version and NEWS for release <NEW VERSION>"
git push
```

14. Update the source on the release management host (or you can check out everything again as step 4):

```
git pull
```

15. Run prepdist.sh to make a tar-ball:

```
# You can cut&paste this:
source prepdist.sh &&
make dist &&
echo "OK" || { echo "FAILED"; cd ..; }
```

16. Extract the tar-ball and make a full build, make sure that the tar-ball and the directory is correctly named.
    If "make check" does not exist, skip it and make the steps manually.

```
tar zxvf <SOFTWARE>-<NEW VERSION>.tar.gz
cd <SOFTWARE>-<NEW VERSION>

# You can cut&paste this:
source ../../prepdist.sh &&
make &&
make check &&
make install &&
rm -rf /tmp/*-release &&
echo "OK" || { echo "FAILED"; cd ..; }
```

17. Retrieve the release tar-ball from the Ubuntu server and put it in distribution repository under source.
18. Sign and checksum the tar-ball using the script sign-distfile.sh (available from the non-public OpenDNSSEC repository).

```
cd <corporation repository>/development/pgp
sh sign-distfile.sh <distribution repository>/source/<SOFTWARE>-<NEW VERSION>.tar.gz
```

19. Verify the signature, first add the public key if you already haven't.

```
cd <corporation repository>/development/pgp
gpg --import opendnssec-pubring.gpg
```

Now verify the signed release.

```
cd <distribution repository>/source
gpg --verify <SOFTWARE>-<NEW VERSION>.tar.gz.sig <SOFTWARE>-<NEW VERSION>.tar.gz
```

20. Commit all distribution files (tar-ball, detached signature and checksum files) to the distribution repository.

```
cd <distribution repository>/source
svn add <SOFTWARE>-<NEW VERSION>.tar.gz*
svn ci -m "<SOFTWARE> <NEW VERSION>" <SOFTWARE>-<NEW VERSION>.tar.gz*
```

21. Close the release and remove the release branch.

```
git flow release finish -m "Tagging <SOFTWARE> <NEW VERSION> release" <NEW VERSION>
git push
git push --tags
git push origin :release/<NEW VERSION>
```