{{lastupdated_at}} by {{lastupdated_by}}

Code release

This page summarizes the steps for a code release. A code release can only be performed by an authorized code integrator.

Making a code release

Creating a code release implies the following steps (in the given order):

- Clone the ctools repository using ssh
- Creation of a release branch from devel branch
- Release testing
- Set release information
- Merging the release branch into the master branch
- Merging the release branch into the devel branch
- Bringing the devel branch ahead of the release branch
- Deleting the release branch
- Tag the release
- Push release in the central repository

Clone the ctools repository using ssh

To start with a code release, first create a clone of the ctools repository using:

```
git clone https://cta-gitlab.irap.omp.eu/<user>/ctools
```

Here, user is your usual user name on the CTA IRAP platform.

Creation of a release branch from devel branch

Now you can step into the ctools directory, switch to the devel branch (just in case you're not already there), and create a release branch:

```
cd ctools
git checkout devel
git branch release
```

Release testing

Using this release branch you will now perform extended code testing. The code testing procedure is still TBW.

Set release information

Before releasing the code, you have to modify a number of files to incorporate release information in the code distribution and to update the code version number. The command sequence for updating release information is:

```
git checkout release
nano configure.ac
  AC_INIT([ctools], [0.4.0], [jurgen.knodlseder@irap.omp.eu], [ctools])
nano ctools.pc.in
  Version: 0.4.0
  Requires.private: GammaLib >= 0.5.0
```

```
nano ChangeLog
nano NEWS
nano README
git add *
git commit -m "Set release information for release X.Y.Z."
```

The nano commands will open an editor (provided you have nano installed; otherwise use any other convenient editor). The lines following the nano command indicate which specific lines need to be changed. Changes in the files ChangeLog, NEWS and README need some more editing. Please take some time to write all important things done, as ctools users will later rely on this information for their work!

Merging the release branch into the master branch

Now you can merge the release branch into the master branch using the commands:

```
git checkout master
git merge release
```

Merging the release branch into the devel branch

We do now the same thing for merging the release into the devel branch. In that way we make sure that the devel branch incorporates all modifications that were made during the code release procedure.

```
git checkout devel
git merge release
```

Bringing the devel branch ahead of the release branch

You now should being the devel branch ahead of the release branch. This means that the devel branch should have an additional commit with respect to the release branch. This is not mandatory, but it assures later that the master branch and the devel branch will never point to the same commit. In that way, git clone will always fetch the devel branch, which is the required default behavior.

To bring the devel branch ahead, execute following commands:

```
git checkout devel
nano AUTHORS
git add AUTHORS
git commit -m "Bring develop branch ahead of master branch after code release."
```

The nano AUTHORS step opens the file AUTHORS in an editor. Just add or remove a blank line to make a file modification.

Deleting the release branch

At this stage you do not need anymore the release branch, and you should delete it now to avoid pushing the release branch to the central repository:

```
git branch -D release
```

Tag the release

Before pushing all modifications to the central repository you need to tag the release using

```
git checkout master
git tag -a ctools-00-04-00 -m "ctools release 0.4.0"
```

We use here an annotated tag with a human readable tag message. Please use always the same format. In the above example, the ctools version X.Y.Z was 0.5.0.

Push release in the central repository

Now you can push all modifications in the central repository using

```
git push --tags https://user@cta-git.irap.omp.eu/gammalib
```

user is again your usual user name on the CTA IRAP platform. The --tags option makes sure that you tag is also pushed towards the central repository.

Now you're done!