



Gaudi software release procedures

Production releases
Bug-fix releases
Development releases



Production Releases

- ◆ **Periodic public, tested, release of all Gaudi and related packages, including**
 - Source code
 - Libraries for Linux and WNT
 - Complete documentation (GUG, code, examples)
- ◆ **Latest released version**
 - becomes recommended default for all users
 - used in Gaudi dependency tree

```
$LHCBSOFT/Gaudi/v6
    /v5
    :
    /GaudiSvc/v1
    /GaudiExamples/v6
        /v5
        :
    :
    :
```

```
GaudiExamples v6 uses GaudiSvc v1
GaudiSvc v1 uses Gaudi v6
```



Bug fix releases

- ◆ **Done whenever bugs need to be fixed**
 - Only for the specific package that needs to be fixed
 - No added functionality relative to major release
 - Tagged with a minor version number
 - e.g. GaudiSvc v1r1
 - Release includes source code and libraries in \$LHCBSOFT
 - `$LHCBSOFT/GaudiSvc/v1r1`
 - `/v1`
 - Other released packages not modified
 - Must be used explicitly in your requirements file:
 - `use GaudiSvc v1r1`
 - CMT knows minor version is compatible with major version, so does not complain if other used packages are in turn using v1 version



CVS repository:

- ◆ On CVS, bug fixes are applied to a *branch*
 - When a bug is fixed, it should be committed to the bug fix branch AND to the head revision, together with updated /doc/release.notes
 - e.g. all GaudiSvc v1 fixes should go to v1r0 branch
`getpack GaudiSvc v1r0, make xyz fix, cvs commit -m 'xyz bug fix'`
 - It is responsibility of developer to also put the bug fixes on the head revision
`getpack GaudiSvc v1 head, make xyz fix, cvs commit -m 'xyz bug fix'`
 - When it is decided to make a bug fix release, librarian tags the branch with the release tag
 - e.g v1r1, v1r2 etc.
- | | |
|----------------------------------|----------------|
| h1 - h2 - v1 - h3 - h4 - h5 - v2 | main branch |
| / / | |
| v1r0 - v1r1 - v1r2 | bug-fix branch |



Development releases

◆ \$LHCBDEV area in AFS, parallel to \$LHCBSOFT

- On regular basis (whenever there is something new), head revision of packages is tagged and built in this area

- For both NT and Linux
- Tag reflects the date of the build

➤ e.g. today's tag is h000407

■ Directory structure:

```
$LHCBDEV/Gaudi/h000407
```

```
h000405
```

```
h000329
```

- If build is successful and works, logical link is made with name of current release:

```
$LHCBDEV/Gaudi/v6->h000407
```

- N.B. Developers may (should!) commit to head revision whenever they have something new that (at least!) compiles

➤ Including updated [release.notes](#)

- Dev release (tag+build) done on demand by librarian

- May be automated in future ('daily build')

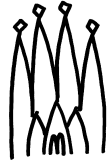


Linking to the DEV area

- ◆ **If you want latest version of everything**
 - **On Unix:** `setenv CMTPATH $HOME/mycmt:$LHCBDEV`
 - **On NT:** add a second path to registry key
`HKEY_LOCAL_MACHINE/SOFTWARE/CMT/path`
 - **Logical links in \$LHCBDEV area ensure you get latest working (undocumented!) versions without changing requirements**
 - (just keep using latest released version)

- ◆ **If you want specific version of just one package**
 - **On Unix:** (e.g. for h000407 version of GaudiSvc)

```
cd $HOME/mycmt
mkdir GaudiSvc
cd GaudiSvc
ln -s $LHCBDEV/GaudiSvc/h000407 v1
```
 - **On NT, since logical links are not possible:**
 - `getpack GaudiSvc h000407` into your CMT directory, build it, and rename directory



When to use the DEV area

- ◆ **NOT if you are just using GAUDI**
 - Use latest released version
 - Use DEV version of any single package for any unreleased features you need to use
- ◆ **NOT if you need to modify a package**
 - Use getpack and modify in your CMT area
- ◆ **In all other cases, preferable to link to the DEV area than to make a private build of head revision**
 - You will be working in same environment as other developers
 - You will get the most recent working modifications
 - You will be actively testing the most recent modifications...



Summary of rules for GAUDI CVS packages

- ◆ **Anyone can commit**
 - To the head revision for new features
 - To the bug fix branch AND the head revision for bug fixes
- ◆ **Commit often, BUT**
 - Discuss your changes beforehand with the GAUDI team
 - Make sure what you commit is coherent and least compiles
 - Give a meaningful comment in -m 'message' field
 - Make sure the changes are documented in release.notes
 - Tell the librarian what changes you have committed
- ◆ **You should never need to tag what you commit**
 - Ask the librarian to apply a bug fix tag or a daily tag