New Structure in CVSROOT and release area

- Outline
  - introduction
  - LHCbSOFT, LHCbDEV, mycmt organization
  - LHCbSOFT and LHCbDEV write access
  - package format
  - How, When
  - the procedure
  - conclusion

Introduction

- LHCbSOFT does not reflect CVSROOT
  - on CVSROOT packages belonging to a group are stored below a “hat”
    - CaloSoft, L1, Velo, Tracking,…
  - “official” packages have no “hat”
    - simgeom, recrich, Gaudi, LHCbEvent,…
  - on LHCbSOFT, LHCbEvent or mycmt/ it is a flat organization:
    - packages are stored without their “hat”
- With the increasing number of packages this flat organization will be difficult to maintain.

proposal

- Every package should belong to a group
- The group name will be kept on CVSROOT, LHCbSOFT, LHCbDEV and mycmt/
  - LHCbSoft/
    - EXTPACK, SICB, GaudiRoot, L1, Velo, L0, CaloSoft, Tr, ITr, OTr
    - SICB/
      - SICBDST, SICBMC, events, simgeom, simguse, recevt, recrich, Finclude, …
      - simgeom/
        - v1, v2 ,v2r2, …
    - GaudiRoot/
      - Gaudi, GaudiAlg, GaudiSvc, LHCbEvent, SicbCnv, HbookCnv, DetDesc,…
      - Gaudi/
        - v1, v2, v3, v3r1, ...

Proposal(2)

- To check out a package:
  - getpack Velo/VSicbCnv v2r1
- To use a package
  - use VSicbCnv v2r1 Velo
- To check out a Gaudi package:
  - getpack GaudiRoot/HbookCnv v6
- To use it
  - use HbookCnv v6 GaudiRoot

Proposal(3)

- LHCbSOFT access
  - readonly is granted to everybody
  - write is granted to librarians only
- LHCbDEV access
  - readonly is granted to everybody
  - write access is granted to package group managers and librarians:
    - each group should have a manager who will have write access on LHCbDEV/groupname
      - Bruce Hay will have write access to LHCbDEV/L1 and LHCbDEV/Velo

Package format

- Gaudi requirements files are now very clumsy:
  - few people understand them
- CMT does not impose any package structure but gives tools to create and use patterns
- The use of patterns allow to predefine some macros used by any packages.
- To create patterns we need a common structure

patterns

- Global patterns
  - are applied by default to all packages
    - include_path none
    - package_stamps
    - package_tag
  - it is possible to ignore a pattern which is not relevant for a package
    - ignore_pattern package_stamps
- patterns
  - are applied on request in the requirements file
    - apply_pattern ld_library_path
    - apply_pattern packageDir
    - apply_pattern package_Cshlibflags
- cmt show patterns

Package format (2)

- branches of pckA : doc/, src/, pckA/, cmt/
  - cmt/ is the new name of mgr/
    - for the moment both names are accepted but in future releases of CMT it is not sure.
  - src/ contains the code *.F , *.cpp
    - a substructure is allowed if absolutely necessary
  - pckA/ contains the include files *.inc, *.h
    - #include “pckA/file.h” no subdirectory
- package name : must start with the group name to avoid clashes
  - CMT will no distinguish between Velo/Event and L1/Event
    - Velo/VeloEvent and L1/L1Event
  - It will be easier to retrieve a package if we use some naming conventions.
- Group name : should be short and easy to recognize to whom it belongs.

Library types

- In Gaudi we create several types of libraries:
  - component library, Library library, static library
- Each type should have a name derived from the package name:
  - The Library library should be called as <package>Lib
    - library VeloEventLib ../src/VeloCLIDS*.cpp
  - the component library should be called as <package>
    - library VeloEvent ../src/VeloEventInstantiation.cpp ../src/VeloEventDll.cpp
  - the static library should be called as <package>Base
    - library SicbCnvBase ../src/static/*.F ../src/static/*.cpp

Component Package format

- Component package
  - include files are internal -> stored in src/
- HbookCnv
  - src, cmt, doc
- requirements file
  - global patterns are set by default
    - package_stamps
    - packageDir
  - apply some patterns relevant for component libraries
    - package_Clinkopts
    - packageCShr
    - package_Cshlibflags

```
package HbookCnv
version v7
branches src mgr doc

use Gaudi v8 GaudiRoot
use CERNLIB v* EXTPACK
# build the component library
library HbookCnv ../src/*.cpp \
                 ../src/*.F

# define component library link options
apply_pattern package_Clinkopts
apply_pattern packageCShr

#==============================
private
apply_pattern package_Cshlibflags
```

Library package format

- Library package
  - exports include files stored in <package> subdirectory
  - builds a Library library and often a component library
- VeloEvent
  - VeloEvent, src, doc, cmt
- requirements file
  - builds VeloEventLib and VeloEvent
  - global patterns are applied:
    - VeloEventLib_stamp is appended to VeloEvent.stamps
  - apply patterns for Library and component libraries if available.
    - package_Llinkopts, ld_library_path, package_Cshlibflags
  - macro VeloEventLib_shlibflags has to be built by the developer

Library package requirements

```
use GaudiSvc v3 GaudiRoot
use GaudiAlg v1 GaudiRoot
use LHCbEvent v7r1 GaudiRoot
include_dirs ${VELOEVENTROOT}
#Library library
library VeloEventLib      ../src/VeloCLIDs.cpp
#component library
library VeloEvent         ../src/VeloEventInstantiation.cpp ../src/VeloEvent_dll.cpp
apply_pattern package_Llinkopts
macro_append VeloEvent_stamps " $(VeloEventDir)/VeloEventLib.stamp"
apply_pattern ld_library_path

# ==============================
private
macro VeloEventLib_shlibflags "$(VeloEventDir)/libVeloEventLib.a"\
VisualC“$(VeloEventDir)/libVeloEventLib.lib"
macro_append VeloEventLib_shlibflags “$(libraryshr_linkopts) $(Gaudi_linkopts) \
$(CLHEP_linkopts)"

apply_pattern package_Cshlibflags
```

How? When?

- Groups should be created as soon as possible
- New packages should adopt the new structure
- GaudiSys/v6 will be reconstructed on a different area using the new structure
  - During this period CVSROOT should not be updated, only bug fixes.
- Because there is no common format in the present release it is difficult to apply an automatic procedure.

The procedure

- Automatic procedure
  - Getpack head revision of all packages
  - remove all CVS/ directories
  - create a cmt/ in every packages
- manual procedure
  - move *.cpp in src/ and *.h in <package>/ paying attention to the current structure,
  - decide to keep/remove subdirectories in src/
  - remove empty branches
  - build the requirements file using patterns
- automatic procedure
  - change all include statements since files have been moved

Procedure (2)

- Build
- Check
- when everything works as in $LHCBSOFT
- create a new CVSROOT
- import all packages
- set LHCBSOFT to the new build area
- keep old CVSROOT as CVSROOT_old
- keep old LHCBSOFT as LHCBSOFT_old

Conclusion

- It is essential that packages adopt a unified format
  - to use CMT patterns
  - to ease navigation
- GaudiSys/v6 in the new format should be available within 2 weeks after the release.