CMT Problems

Please report here, as precisely as you can, any general or specific problem you have with the use of CMT. Some of the problems here will probably be redeclared later as JIRA items. PLEASE GIVE YOUR NAME AT THE END OF YOUR CONTRIBUTION.

Container packages and directories

From R.Giannitrapani

One of the problem encountered in MRvcmt is the following: I'm using the command "cmt show packages" to have the list of the packages contained in the cmtpaths. If some packages belongs to a subdirectory of the cmtpaths and this subdirectory is not a package itself, such contained packages are not shown in the output of this command and so will not shown in the tree of MRvcmt.

This is not a bug of CMT, I think it is a feature; but sometime it could be helpful to just have directories that can contains packages without the need for them to be a true cmt package.

In general this is due (imho) to the fact that in CMT container packages have two roles:

- on one side they are just container; actually, if you use a contained package you will write in your requirements file something like "use Geant4 v2r6* IExternal" and you never specify a version for "IExternal", i.e. you are using it exactly as a directory, not as a package

- on the other side they need to be packages, so they need to have the normal package structure (cmt directory and so on), to be seen from cmt (and so from the current version of MRvcmt).

This double role can create more subtle effects, for example if you have two versions of a container package, as I've tried to describe here http://www.fisica.uniud.it/~glast/MRStudio/history.html (in fact I'm tring something different from MRvcmt in the incoming MRStudio).

Last problem (that is also reported in some JIRA issues cited below), the command of cmt to show packages has some problem that prevent to show all the packages contained in a container package (and again this is reflected in an incomplete tree of MRvcmt .. also for this, MRStudio will not use cmt to build the packages tree, but it will use a paths traversal search with regular expressions).

From D.Chamont

I feel the container packages was originally meant to refer to others packages at the same "directory level", not as subdirectories of the container package itself. There was no need for such double role. On other hand, there is this concept of subpackages in CMT which is far from documented, and need investigation.

The bug CMT-38 with Geant4 mentionned in JIRA seems fixed in v1r18. Yet, the overall problem of "cmt show packages" remains : it should be able to parse subdirectories even when they are not packages, and even if the super-package has not been checked out. At least subdirectories shoudl be parsed to a configurable depth, so that "cmt show uses" is consistent with "cmt show packages".

From D.Chamont

I have the feeling there is no more problems with v1r18p20060301.
Please check.

CMTCVS Pluggin

From D.Chamont

The "cmt cvstags <package>" command is used in several key places :

- We suspect CMT itself uses this to determine a version-folder name to use for the "cmt co" command, when a specific version is not specified.

- MRvcmt uses it to form a list for the pull down menu in the checkout dialog.

- The Release Manager use the first tag as the one to be used so to build LATEST.

- Some other developers (Jim) use it to trace the packages tagging.

On top of few bugs, there is much confusion about what should be the ordering of tags displayed by thsi command. This is what the CMT team expect the pluggin to do :

- The first tag, internally called the top tag, should be the one which has the higher CVS revision number. I also feel the tags on branches should not be eligibles, but I am not sure this is what the authors wanted. If so, this does not work.

- The following tags comes in chronological order, as they appear in file "<package>/cmt/requirements,v" . I feel the authors tried to filter out the branch tags, again without success, at least for repositories where the branch revision number contains an additionnal magic 0 just before the end.

After some unsatisfying debate with CMT team, where I did not succeed to understand what the pluggin was meant to do, I found no way to avoid a direct modification of the pluggin, and did the following :

- Modification of function check_newer() so that the selection of the top tag ignore the tags on CVS branches.

- Modification of the function getting the list of all tags, so that branch tags are ignored.

Until now, I do not want to prevent users from setting their own private tags, and yet take profit of "cmt co -R", so I didid modified the pluggin so to filter out whatever is not an official GLAST tag (v* or v*r* or v*r*p*). The tools such as the Release Manager still have to do this filtering by themselves.

Installation Area

Can somebody explain why this CMT feature has been discarded ?

From Heather

I'm curious to see what others recall, but here are some notes I've found:

- Here is a page that Traudl maintained concerning the Install Area: http://www.slac.stanford.edu/~hansl/soft/TestInstall/
- When using the Install Area, there were problems using CMT packages installed on NFS space - often the libraries could not be found. Apparently Traudl found some work-around, but I know I and others seemingly randomly continued to have trouble.
- From the core minutes April 20, 2004: *At least one other issue remains to be resolved: what to do about the xml file hierarchy in xmlGeoDbs. The normal install area expectation is that data files are stored in a single level.*
  This would definitely be problematic as we do need some structure for the XML files since we support multiple configurations
- Toby disabled the install area by updating GlastPolicy on Oct 10, 2004 - perhaps he recalls exactly what was the major problem at the time that forced him to do this - though I'll point out..I don't recall many people were upset to see the Install Area disabled.

From Joanne Bogart

I recall having problems with picking up the right version of libraries. It's been a while now so my memory is suspect, but I think that old libraries in the Install Area maybe didn't get deleted at the expected time and would get used preferentially. Or something like that. However, I never encountered any such problems with Navid's implementation of a similar functionality, so I don't see any reason to attempt to revive the official CMT Install Area.

CMT runtime execution

From David Chamont

I remember to hear often people (toby, navid, richard ?) complaining about the <package>_ROOT variables generated by CMT. What is the reason for this ? There are too much variables declared by CMT ?

CMT and Windows

Several cases are under investigation :

- Does CMT support new Visual Studio (2005 or v8) ? Or is it planned ?
- Is there a way to switch off the generation of Visual Studio files by CMT ?
- Where are stored the temporary files (which are used to trigger the cmtcvs pluggin) and how to customize this ?

CMT Is Too Coarse

From Joanne Bogart

Most of my complaints boil down to a lack of fine control in a natural way. As soon as there is more than one build target in a package, there are likely to be problems. As Navid mentioned, we've had trouble with multiple library targets when one should depend on the other, and trouble in general with multiple targets in a package because often they shouldn't have the same set of use statements. Private/public doesn't do the trick; the only sure way I"m aware of to deal with this is not to have multiple targets in a package unless they have similar or identical properties w.r.t. uses. The CMT use concept is also too coarse with respect to time. A package - actually, a target within a package - might need access to another at run-time but not during compile or link, or at compile and run time, but not link time. There is no way to express this with a use statement.

Careful use of -no_auto_imports and other features (by those who understand them, not a group I belong to) can sometimes help, but such solutions seem to be very fragile, for example causing problems with build order. Overall, there is just too much riding on a CMT use statement. The other tools at one's disposal, like -no_auto_imports, are essentially tinkering, and they tend to interact in unpleasant and unpredictable ways.

From Navid

One of the biggest flaws of CMT (or our way of using it) is that when
you have multiple libraries/applications listed in your requirements
file you can't easily make them depend on each other. Our attempt at
solving this is to use static libraries (which are built before
shared ones) and to separate out test apps into their own packages.

CMT links against unnecessary libraries. When you depend on another
package, you automatically inherit all the libraries that package
makes and all its sub-packages. Again this is not desired all the
time. A variation of this is that if your package creates two
constituents and one depends on a different package and the other
does not. Both your constituents have no choice but to use this other
package.

There's more flaws in CMT (redundant versioning, container packages,
etc..) but the above two are recent ones that are biting us over and
over again that are also major flaws I think.

From Riccardo

... for example, I see a package as an atomic object in CMT design .. I think that if in the same package two constituents uses different dependenecies, this is not CMT problem but point to a need of reorganization of the packages, maybe these two constituents should not really belong to the same package (but again, I'm not an expert on this, so maybe I'm completely wrong)...

From David Chamont

I also heard about problems with the huge number of useless libraries which
are propagated to the final applications. The art of "no auto_import" can
help us to reduce the problem, but not easy enough (see FluxSvc).

CMT Items in JIRA

key	summary	assignee	status
Unable to locate Jira server for this macro. It may be due to Application Link configuration.			

Error rendering macro 'jiraissues'

Macro params are invalid