I just collected some emails which address the compiler flag issue:

If you want one package to compile in dbg (but the rest in opt) add

macro_append (right macro) " -g"

for -g it would be:

macro_append cppflags " -g"

to your requirements file.

If you want the O2 flag removed (or any other) add

macro_remove (right macro) " -O2"

to your requirements file.

To find out which macro a flag is in read through the mail by RD:

Hi Andreas,

macro_remove works. The question is to find which macro.

From any makefile, e.g. .make you'll see that the compile command is cppcomp

% cmt show macro cppcomp

Package CMT v1r16p20040701 defines macro cppcomp as '$(cpp) -c $(includes) $(cppdebugflags) $(cppflags) $(pp_cppflags)' for default tag # # Selection :

cppcomp='$(cpp) -c $(includes) $(cppdebugflags) $(cppflags) $(pp_cppflags)'

so it is one of the above macros, not necessarily cppflags.

Apparently it is in cppdebugflags:

% cmt show macro cppdebugflags

Package

`GaudiPolicy`

v5r15p1 defines macro cppdebugflags as '$(cppoptimized_s)' for tag 'optimized' # # Selection : cppdebugflags='$(cppoptimized_s)'

% cmt show macro_value cppdebugflags -O2

see you, RD

Hi

RD is right to say that the "The question is to find which macro"

To complete this info, generally to obtain this information you can do :

> *cmt show macros | grep "O2"*

This should give you something like:

> *cmt show macros | grep O2*
cppoptimized_s='-O2' foptimized_s='-O2'

Telling you that either cppoptimized_s or foptimized_s is providing this option [according to whether you are using C++ or Fortran]

Then of course you can obtain more details about this macro (especially WHO defines it):

> *cmt show macro cppoptimized_s*
# Package GaudiPolicy v5r15p1 defines macro cppoptimized_s as '-O2' for default tag # # Selection :
cppoptimized_s='-O2'

Then you may freely modify the definition of this macro in you package using a macro_remove, a macro_append, a macro_prepend, etc...

Remember also that if the modification is done privately it won't influence your client packages

========== private macro_remove cppoptimized_s '-O2'
==========

while if made in a public section of your package, the mod will affect all client packages.

You can also figure out how this macro is effectively used:

> *cmt show macros | grep '$.cppoptimized_s'*
cppdebugflags='$(cppoptimized_s)'

% cmt show macros | grep '$.cppdebugflags' cppcomp='$(cpp) -c $(includes) $(cppdebugflags) $(cppflags) $(pp_cppflags)'

% cmt show macros | grep '$.cppcomp'

Here cppcomp is directly used in the make fragment generated by CMT

% grep cppcomp $CMTROOT/fragments/* .../cpp: ... .../cpp_library: ...

Cheers Christian.

-- DerSchrecklicheSven - 17 Nov 2004

This topic: Main > CompilerFlags
Topic revision: r1 - 2004-11-17 - AndreasWildauer