



The Implementation of a Extensible PhysicsListFactory

Robert Hatcher
Fermilab

Geant4 Collaboration Mtg 2015-10-01 Parallel 5A

Usage (old)



- Examples of use in examples/tests
 - two forms: given a string or from the \$PHYSLIST env variable
 - new factory is “drop in” compatible (currently in g4alt namespace)

```
const char* pname = 0;
if ( argc > 2 ) { //Physics List via command line
    G4cout<<argv[2]<<G4endl;
    pname = argv[2];
}
else { //Search physics list via env.variable
    pname = getenv("PHYSLIST");
    if ( !pname ) pname = "local";//No command line and no env.
}
if ( strcmp(pname,"local")==0 ) {
    runManager->SetUserInitialization(new BeamTestPhysicsList);
}
else {
    G4PhysListFactory factory;
    runManager->SetUserInitialization( factory.GetReferencePhysList(pname) );
}

```

```
// return FTFP_BERT unless $PHYSLIST env set
// no means of local choice of default
G4PhysListFactory factory;
runManager->SetUserInitialization(factory.ReferencePhysList());

```

Involved Classes



- G4PhysListFactoryAlt [.hh | .cc]
 - defines g4alt::G4PhysListFactory
 - implements extended version of existing G4PhysListFactory
 - factory just acts as a dispatcher to G4PhysListRegistry
- G4PhysListRegistry [.hh | .cc]
 - singleton holding strings and their mapping to G4PhysListStamper
 - no internal pre-defined list; external code can self register
 - also hold extension mapping (e.g. _EMX etc)
- G4PhysListStamper.hh
 - PhysList equivalent of G4PhysicsConstructorFactory
 - clash of terminology “factory” as individual builder or collection
- G4RegisterPhysLists.cc
 - register all G4 supplied lists
 - special handling for physics lists because they’re templated; no .cc
 - user physics lists can similarly register themselves with the factory



How to Extend

- Examples of new lists and new EM extension
 - for supplied standards make changes to these files
 - for user extensions, they can do it in code compiled into their own library

```
Index: src/G4PhysListRegistry.cc
=====
--- src/G4PhysListRegistry.cc    (revision 93012)
+++ src/G4PhysListRegistry.cc    (working copy)
@@ -65,6 +65,7 @@
     theInstance->AddPhysicsExtension("EMZ","G4EmStandardPhysics_option4");
     theInstance->AddPhysicsExtension("LIV","G4EmLivermorePhysics");
     theInstance->AddPhysicsExtension("PEN","G4EmPenelopePhysics");
+    theInstance->AddPhysicsExtension("GS","G4EmStandardPhysicsGS");

     return theInstance;
 }
Index: src/G4RegisterPhysLists.cc
=====
--- src/G4RegisterPhysLists.cc  (revision 93012)
+++ src/G4RegisterPhysLists.cc  (working copy)
@@ -118,5 +118,7 @@
 #include "G4GenericPhysicsList.hh"
 G4_DECLARE_PHYSLIST_FACTORY(G4GenericPhysicsList);

+#include "QGSP_BIC_AllHP.hh"
+G4_DECLARE_PHYSLIST_FACTORY(QGSP_BIC_AllHP);
```

g4plfactory_test38



- Implemented a stand-alone “test38”
 - not integrated into full ctest chain
 - exercise the interfaces
 - provides optional cross check with old implementation

```
rhatcher:test38_gcc-c++11 $ ./g4plfactory_test38 -h
./g4plfactory_test38: G4PLFactoryTest - a simplified Geant4 app for testing
the G4PhysListFactory
  ./g4plfactory_test38 [options] [physList1 physList2[=N]]
-h --help          this output
-f                print phylist factory status
-F               print old phylist factory availability
-c              print physics ctor list
-r              print physics list registry list
                repeat to print before adding 2nd library
-v --verbose      increase program verbosity
-V <n>           set factory verbosity
-D --defaults    add default tests even if user supplied tests
-o --old         test old factory
-e --env=PNAME   PhysicsList to use as env variable [QGSP_BERT]
                use "skip" to skip these 2 tests
--lend          try ShieldingLEND (needs special data) in default list
--xyzy         try to add non-existent physics ctor in default list
                (will though throw G4Exception w/ --fatal)
--fatal        throw exception if new factory can't satisfy request
```

If given, the list of physics lists to try override the default set.
User can specify if they expect each to work with the
new (1), old (2), both (3) or neither (0) factory;
if unspecified, assumes 3.

g4plfactory_test38



- -f = factory status
- recommend to print this if user requests non-existent combination
- simply call:

PrintAvailablePhysLists()

Base G4VModularPhysicsLists in G4PhysListRegistry are:

```
[ 0] "FTFP_BERT"  
[ 1] "FTFP_BERT_HP"  
[ 2] "FTFP_BERT_TRV"  
[ 3] "FTFP_INCLXX"  
[ 4] "FTFP_INCLXX_HP"  
[ 5] "FTF_BIC"  
[ 6] "G4GenericPhysicsList"  
[ 7] "LBE"  
[ 8] "MyPL0" ← added by compiled object linked to executable  
[ 9] "MyPL1" ← added by shared library linked to executable  
[10] "MyPL2" ← added upon dynamic library load at runtime  
[11] "NuBeam"  
[12] "QBBC"  
[13] "QGSP_BERT"  
[14] "QGSP_BERT_HP"  
[15] "QGSP_BIC"  
[16] "QGSP_BIC_HP"  
[17] "QGSP_FTFP_BERT"  
[18] "QGSP_INCLXX"  
[19] "QGSP_INCLXX_HP"  
[20] "QGS_BIC"  
[21] "Shielding"  
[22] "ShieldingLEND"  
[23] "ShieldingM"  
[24] "mys::MyNSPL3" ← allow for user namespaces
```

example user addition ←

example:

"mys::MyNSPL3_LIV+ALTDK+NEWPHY"

starts with user supplied physics list
replaces standard E&M w/ Livermore variant
and adds new decays and exotic physics

Replacement mappings in G4PhysListRegistry are:

```
→ ALTDK => G4NewDecayPhysics  
EMV => G4EmStandardPhysics_option1  
EMX => G4EmStandardPhysics_option2  
EMY => G4EmStandardPhysics_option3  
EMZ => G4EmStandardPhysics_option4  
LIV => G4EmLivermorePhysics  
→ NEWPHY => mys::G4NewExoticPhysics  
PEN => G4EmPenelopePhysics  
→ XYZZY => NoSuchPhysics [unregistered physics] ←
```

Use these mapping to extend physics list; append with _EXT or +EXT to use ReplacePhysics() ("_") or RegisterPhysics() ("+").