



# Extensible PhysicsList Factory Status and Update



**GEANT4**  
A SIMULATION TOOLKIT

Robert Hatcher

Geant4 2018 Collaboration Mtg - Parallel 6A:

open topics in physics lists and validation tools

30 August 2018

# g4alt::PhysListFactory Status ...

Complete: does exactly what it says on the box.

Works for all build setups:

- unix vs. windows

- shared libraries vs static libraries

- cmake global vs. gmake granular libs

See my talk in the parallel 6B session on extensibleFactory example.

Outstanding question: Swap current one to “legacy” and swap this one in without the namespace `g4alt`?

Short talk ... huh?

# So what's the issue...

The design is tangled due extra complications from static libraries and the split of granular libraries for the physics constructors and a touch of to windows specific bits ... much of this was covered previously:

WG Mtg - 4 Dec 2017

<https://indico.cern.ch/event/685094/contributions/2814588/>

WG Mtg - 5 Feb 2018

<https://indico.cern.ch/event/701245/contributions/2885687/>

2<sup>nd</sup> talk concluded with: ...

# So what's the issue...

## Conclusion

- This should satisfy both static and shared library builds, without a circular dependency, even when using “granular” libraries
- Can things still go wrong?
  - Possibly leave PhysCtorRegistry unfilled if all the following are simultaneously true:
    - Using static libraries
    - Using PhysCtorRegistry (aka ctor factory)
    - Not using PhysListRegistry (aka extensible factory)
    - And not explicitly referencing a standard PhysList
    - Non-standard PhysList doesn't use self-registration



# Huh?

Turns out there are such cases... [deep sigh]

2018-02-27 ... after my commits to resolve all the issues I talked about at the previous WG meeting:

Windows build of CDash tests fail

Investigated and found issues w/ `example-ext-medical-dna-*`

Technically not a *PhyListFactory*, ... but ctor factory issue

# “Fix”

Rearranged ctor code registration some; requested permission to change the example by adding:

```
#include "G4SystemOfUnits.hh"
```

```
#include "G4EmDNAChemistry.hh"
```

```
#include "G4PhysicsConstructorRegistry.hh"
```

```
+ // G4RegisterPhysicsConstructors.icc is necessary for static builds
```

```
+ #define REGREF PhysicsList (ensure next line's static vars are unique)
```

```
+ #include "G4RegisterPhysicsConstructors.icc" (ensure ctors are registered)
```

```
#include "CommandLineParser.hh"
```

to

```
/trunk/geant4/examples/extended/medical/dna/chem1/src/PhysicsList.cc
```

```
/trunk/geant4/examples/extended/medical/dna/chem2/src/PhysicsList.cc
```

```
/trunk/geant4/examples/extended/medical/dna/chem3/src/PhysicsList.cc
```

```
/trunk/geant4/examples/extended/medical/dna/chem4/src/PhysicsList.cc
```

```
/trunk/geant4/examples/extended/medical/dna/mfp/src/PhysicsList.cc
```

```
/trunk/geant4/examples/extended/medical/dna/neuron/src/PhysicsList.cc
```

```
/trunk/geant4/examples/extended/medical/dna/spower/src/PhysicsList.cc
```

# ...Almost...

That should have, in principle, fixed everything

But some weird oddity w/ MS Windows and the static variables when using *shared* libraries

Could not figure it out.

In the end, disabled include for all WIN cases

```
#if defined(WIN32) && !defined(USING_STATIC_LIBS)
// WIN32 linker/loader works differently than on unix
// these G4_REFERENCE_PHYSCONSTR_FACTORY macro calls in fact can't be used
// in "shared" link mode. In principle they should be called
// in "static" link mode (that's the point of this file).
// Currently there is no build -D flag that distinguishes the two cases...
// but users can set USING_STATIC_LIBS
#define G4RegisterPhysicsConstructors_icc 1
#endif // WIN32
```

The include isn't needed in case of shared libraries, but does no harm in Unix/Linux cases ... but no available compilation flag

## TL;DR ... Where are we:

Those examples will compile for all cases, and work for all but Windows w/ static libraries.

If one forces a compilation `-DUSING_STATIC_LIBS` flag the last case will also work correctly, but otherwise for that case the ctor factory has no registered physics constructors in it and thus the example's specialized `PhysicList` can't be assembled.

# Options going forward:

Add a compilation flag `-DUSING_STATIC_LIBS` (for WIN case)  
not very satisfying ... kind of fussy ...

A bit more work, but what I'd find "cleaner" is:

- Back out the previous addition to examples
- Change each example to use extensible `g4alt::PhysListFactory`
- Change each example's list name from generic "PhysicsList" to a specific name, e.g. `DNAChem1PhysicsList` (just for clarity)
- Have example's list self-register with extensible factory, see `examples/extended/physicslists/extensibleFactory/extensibleFactory.cc`
  - `#include "DNAChem1PhysicsList.hh"`
  - `G4_DECLARE_PHYSLIST_FACTORY(DNAChem1PhysicsList);`
- Make example's "default" physics list the specific list  
`factory.SetDefaultReferencePhysList("DNAChem1PhysicsList");`