

Status of G4CMP Migration to Geant4 11

Jake Inman (PNNL), Micah Johnson (LLNL),
Michael Kelsey (TAMU), Stefan Zatschler (UofT)

Geant4 Technical Forum, 23 Oct 2025

Outline

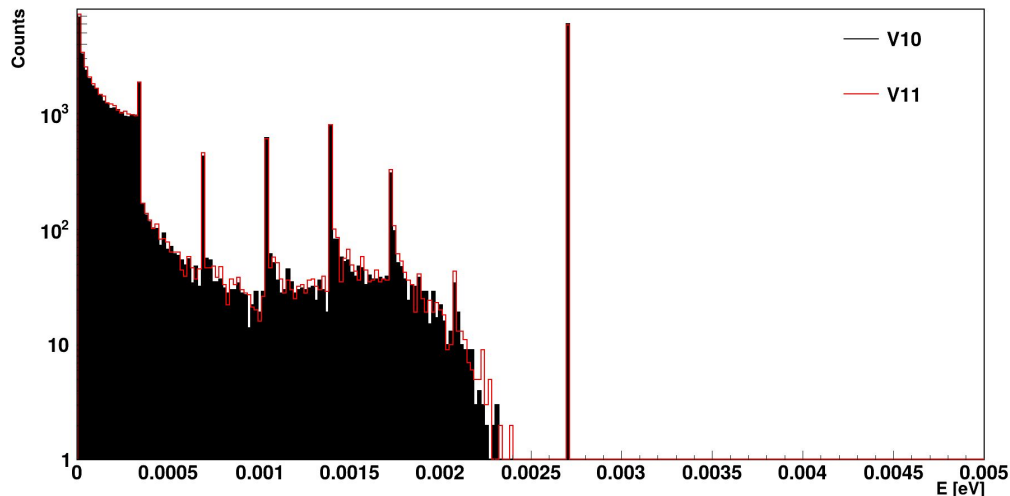
- Background, Current Status
- Only Four Required Modifications
- Blocker and Geant4 Suggestion
- Backward Compatible Option

Background, Current Status

G4CMP has been frozen at Geant4 10.7.4 to support stability of simulations for SuperCDMS. QIS community would like to move forward, without losing their current code investment.

Jake Inman (PNNL) and Micah Johnson (LLNL) successfully migrated G4CMP to Geant4 11

Consistent physics results (phonon energy absorption) between versions (red vs. black at right)



Tracking progress in an internal JIRA ticket, expect to deploy by end of year.

Four Required Modifications

- **Easy:** Replace forward declarations of `G4VTouchable` with `#include` (no longer a class, just a typedef)
- **Medium:** Replace `std::random_shuffle` with `std::shuffle` (implications for random reproducibility)
- **Hard:** Replace usage of `G4String` member functions with new `G4StrUtil` namespace
- **Blocker:** `G4PhysicsModelCatalog` no longer has public registration for custom/user physics

Backward compatible migration may be possible

Adding Custom Physics Models in *G4PhysicsModelCatalog*

- The idea is to keep the property of invariance (of *ID*, *index*, *name*) only for the official Geant4 physics models (*i.e.* those included in the *Initialize()* method), while insuring consistency only – *i.e.* unique ID for any application – for the custom physics models
 - By introducing a new, public method, *RegisterCustomModel(G4String& name)* which the user is responsible to call for each custom physics model
 - And only for those, *i.e.* it must not be called for official Geant4 physics models
 - We store the information of these custom physics models in a separate vector (of strings, *i.e.* their names), using consecutive ID values starting from **40'000**
 - The same custom physics model could have different ID values according to the application, physics list, and Geant4 version, exactly as it was the case for all physics models before G4 11.0
 - This solution would not require a change of users' interface – because we are only **adding one public method** – therefore it could be also included in a patch (possible for any of the versions 11.{0,1,2,3,...})

Backward Compatible Migration

Possible to complete a fully backward compatible migration, but need to decide if choices are acceptable and maintainable moving forward.

- `#include "G4VTouchable.hh"` in place of forward declarations
- `std::shuffle` is available in C++14, so use it, replacing `G4CMP::RandomIndex()` function with functor class
- Either `G4CMPString` wrapper, or `#if` version blocks where needed
- Copy `G4PhysicsModelCatalog.cc` into `G4CMP/library/src/`, surround whole file with `#if` version block, add extra line to model list

Summary and Conclusions

Migrating the G4CMP library to be compatible with Geant4 11 was straightforward, if somewhat invasive

- Only four items needed migration, of which one is an apparent blocker.
- `G4PhysicsModelCatalog::Register()` has already been reviewed by the Geant4 experts and can be restored without compromising the improvements introduced with Geant4 11. *Thank you!*

The full migration may be possible to complete in a backward compatible way

- We will evaluate how to encapsulate the `G4String/G4StrUtil` instances

We appreciate the quick, and supportive, response from Geant4 to our questions, even at this late date

Backup Slides

`std::shuffle` and Random Engines

Under C++17, `std::random_shuffle` no longer available (used in `G4CMPEnergyPartition::DoPartition`), previously just deprecated

Replacement is `std::shuffle`, but requires STL [uniform random bit generator](#) object as argument.

- `HepRandomEngine` [does not provide](#) correct STL interface
- Using `std::mt19937` from `<random>`, doesn't allow G4 seed reproducibility

Simple change: Replace `G4CMP::RandomIndex` function with functor class containing `operator()` that does what function does now

- Could provide [general-purpose wrapper](#) to map `HepRandomEngine` to STL interface; might propose this to Geant4

STL `<random>` vs. CLHEP `HepRandomEngine`

No top-level base class; three templated types with some concrete typedefs

- `std::linear_congruential_engine`
- `std::mersenne_twister_engine`
- `std::subtract_with_carry_engine`

All three types define same set of public member functions

- Static `min()`, `max()` functions which return `unsigned int` range of values
- `seed()`, to specify either seed value or `std::seed_seq` object
- `discard()`, which steps internal state without returning value for use
- `operator()`, which returns next `unsigned int` random value
- operators `<<` and `>>` for stream I/O operations

Use CLHEP's casting operator `unsigned int()` inside `operator()`; don't need to define the whole interface just to pass into `std::shuffle`

G4String and New G4StrUtil

G4String was written before STL existed

- Originally based on RogueWave string class
- Numerous convenience functions for substrings, case changes, etc.

Convenience functions have been moved to `G4StrUtil::` namespace

- Called by passing `G4String` reference (const or non-const)
- Some removed entirely, in favor of direct `std::string` functions

G4 11.2 still had functionality, but generated compiler "deprecated" warnings

Invasive changes, all protected with `#if G4VERSION_NUMBER` blocks

Kelsey is tempted to define a `G4CMPString` class equal to old `G4String`, with call throughs ¹¹

Blocker: G4PhysicsModelCatalog

We register G4CMP physics into catalog with (Geant4 10.7.p04)

```
fPhysicsModelID = G4PhysicsModelCatalog::Register("G4CMP process");  
track.SetAuxiliaryTrackInformation(fPhysicsModelID, ...);
```

Geant4 11 no longer has public interface to add new models or get indices

- Could you restore a public interface (outside of catalog initialization) to generate a model ID and index for use with AuxiliaryTrackInfo?
 - If the catalog wants to tell us what model ID to use, that's fine with us!
- Could copy `G4PhysicsModelCatalog.cc` into G4CMP library, force-load it to override Geant4 version
 - Use `#if G4VERSION_NUMBER` block around contents of file

<https://geant4-forum.web.cern.ch/t/adding-a-model-to-g4physicsmodelcatalog/11780>

<https://geant4-forum.web.cern.ch/t/version-11-compatibility-physics-model-catalog/14558>

Next Steps, Planning Deployment

Will create **G4CMP-494** feature branch in main repository, merge fork there

- Resolve questions about migration details
- Requirement for `G4String/G4StrUtil` with preprocessor blocks would make ongoing development more difficult, prone to mistakes
- Should we make this G4 11 *only*, or use "`GCMPString`" wrapper?

Deploy at end of 2025 or early 2026 (after Geant4 11.4 release, maybe after Winter Break)

- If backward compatible option is chosen, *could* deploy early, giving end users more flexibility around migrating their applications

Input and suggestions in [G4CMP-494](#) JIRA ticket are encouraged!