https://gitlab.cern.ch/adotti/Geant4LoggingMonitoring

New Logging Monitoring Example

Performance Made Easy

Andrea Dotti (adotti@slac.stanford.edu) ; SLAC/SD/EPP/Computing

Geant4 21st Collaboration Meeting – Ferrara, 12-16 September 2016

U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

Introduction

- When we see changes in CPU performances the first question we ask ourselves is: “did the physics change? Do we see less/more tracks/steps?”
  - Soon provides answers to these questions via detailed profiling of specific applications
- The same question is also important for physics validation. While a change of number of tracks does not teach us much per-se is an piece of information
  - In this case one needs to manually change the user action code, add counters and print them out
- Starting from 10.3 we will allow multiple user-actions in each job (see kernal updates talk in plenary 7)
  - We can now think to develop some common utilities that can be used to monitor these quantities in a general case
  - We can share this code so anybody can use the same code to monitor applications

The LoggingMonitoring Example

SLAC

A set of “monitors” has been created to allow one or more of:

- Number of steps
- Number of tracks and time needed to complete
- Number of events and time needed to complete simulation
- Number of runs and time needed to complete simulation

Filters can be associated to each “monitor”:

- Filter on particle type
- Filter on energy window
- Filter on geometry volume

Allows for logging to:

- Standard Output
- Histograms (via G4Analysis)
- Ntuples (via G4Analysis)

User Interface

Two user interfaces provided:

- UI commands:
  1) Instantiate monitor in G4ActionInitialization class (this will automatically use the new multiple-action functionality)
  2) Definition of monitor, filters and loggers is fully UI command based
- Via C++ API for example to create advanced filters

G4ActionInitialization::Build()

SLAC

```
void MyActionInitialization::Build() const {
    // your user code goes here
    [...]
    // these are the last two lines:
    auto monman = new G4MonitorManager;
    G4AutoDelete::Register( monman );
}

//Same for BuildForMaster
```

UI commands

```
/run/initialize
# Count number of steps of e-
# with 10MeV<E<100MeV in volume “Calo”
/monitoring/step/create myMon1
/monitoring/step/addParticleFilter e- myMon1
/monitoring/step/addEnergyFilter 10 100 myMon1 //in MeV
/monitoring/step/addLogicalVolumeFilter Calo myMon1

# Save info in histograms (binning is automatic):
/monitoring/setOutput Histo file.root


/monitoring/initialize
/run/beamOn 100
/monitoring/finalize
```

Example

Proposing a new example in:

- extended/runAndEvent/Monitoring
- Simple geometry (box of material) with trivial user actions

Showing both UI monitoring ad C++ API

Status

SLAC

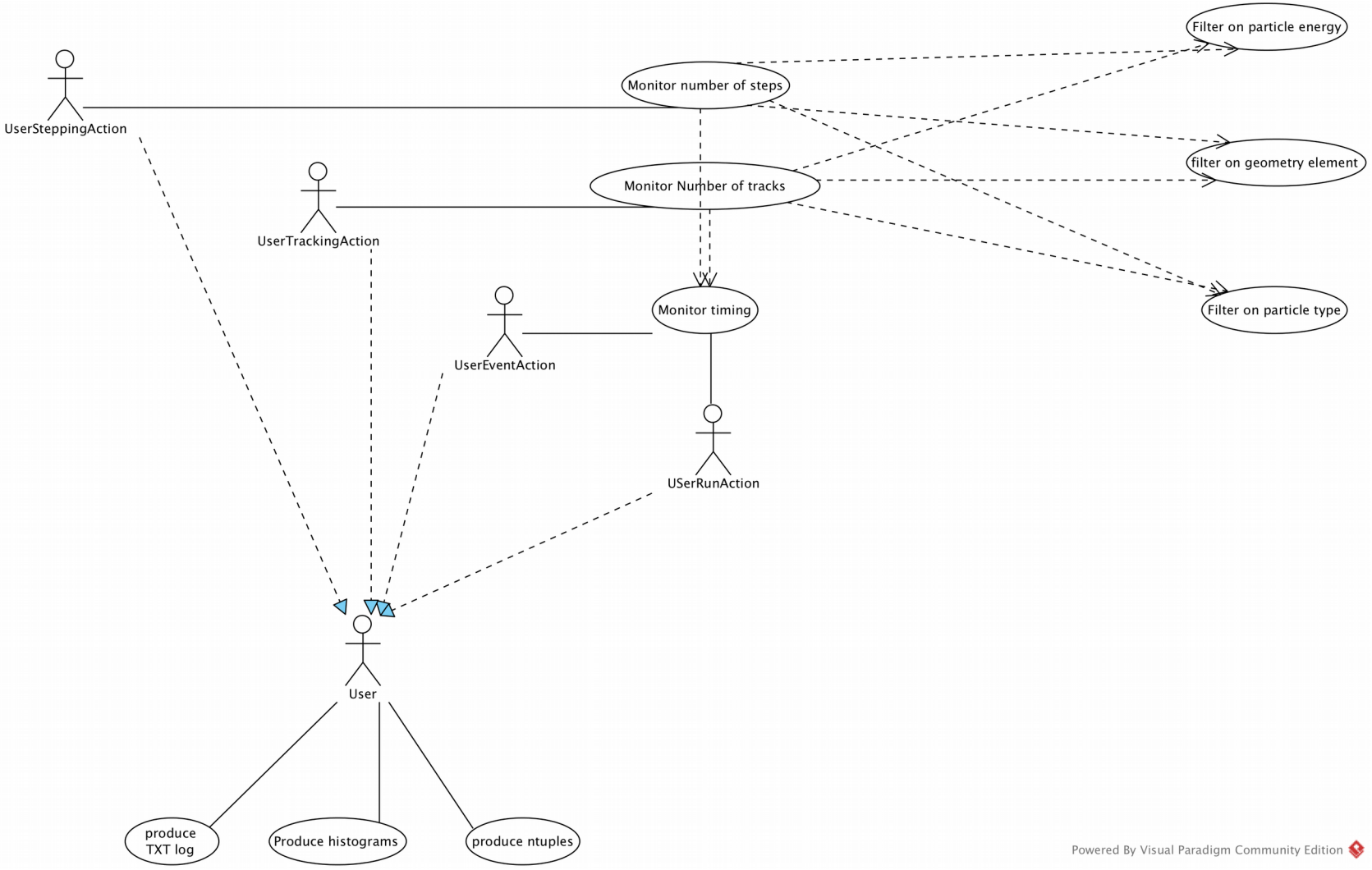
Code is ready since some time, however we never included in SVN for two reasons:

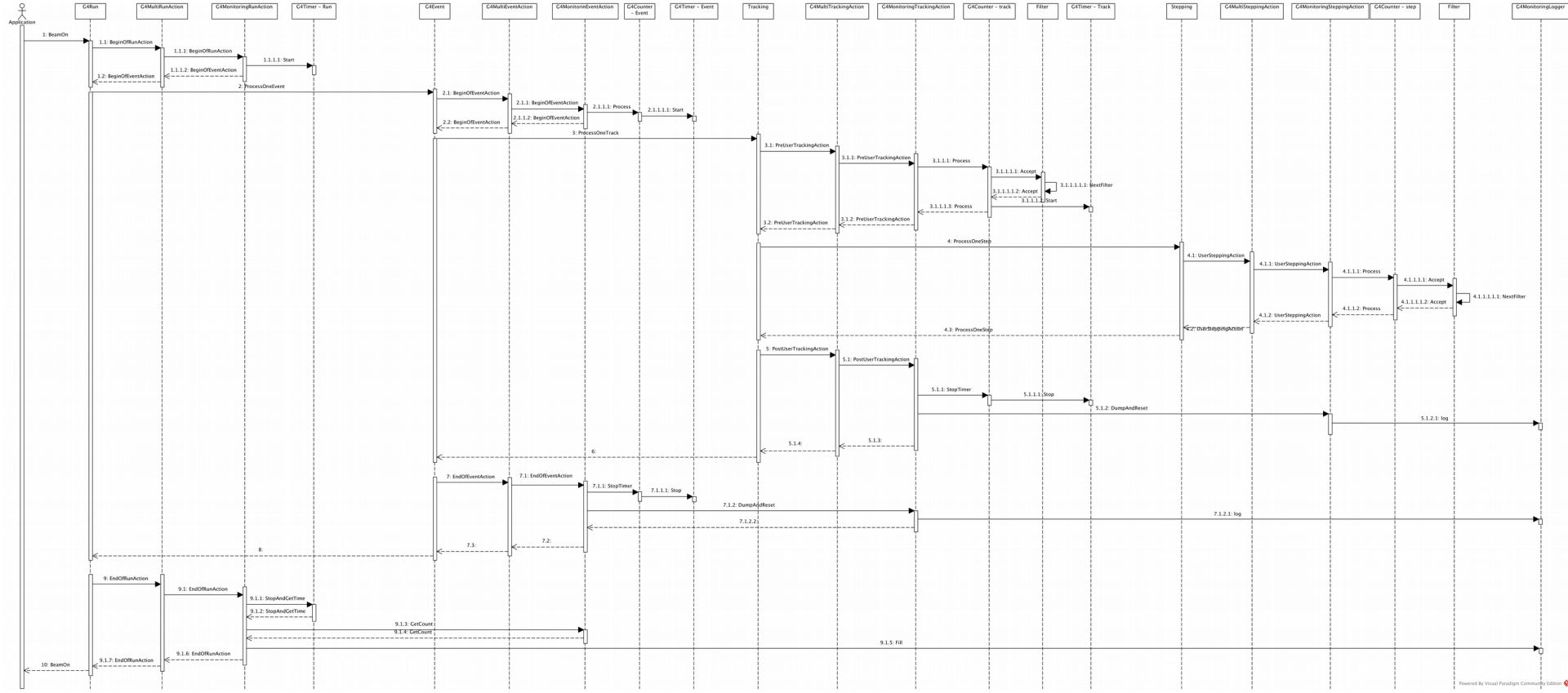
- We first needed the possibility to have multiple-user actions
- Run category will depend on G4Analysis (for histograms and ntuples): discussion please!

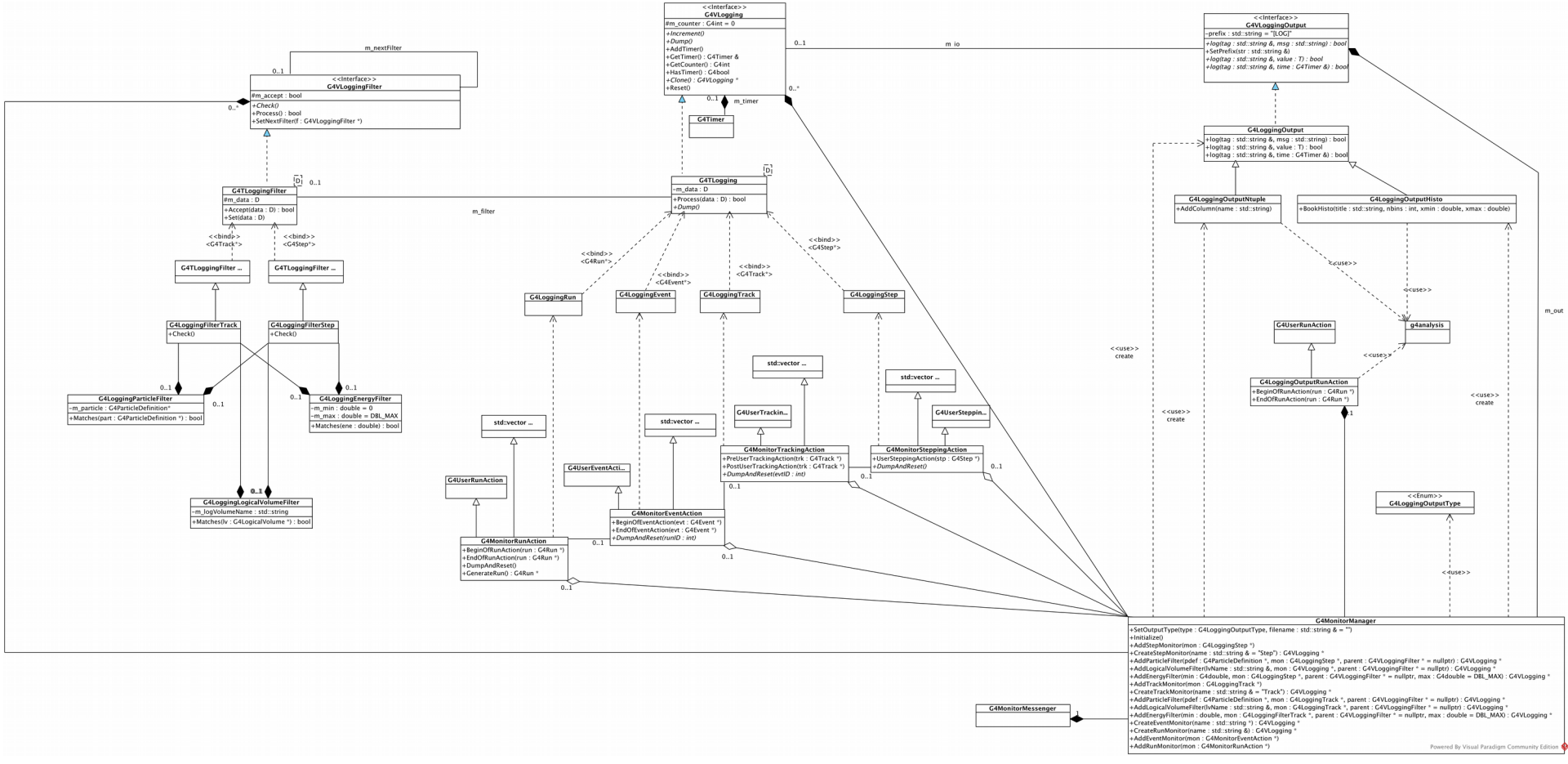
Aiming at public release for 10.3

- Proposing to then use this functionality for monitoring int the context of performance monitoring

Backup


Filter on particle energy
Monitor number of steps
UserSteppingAction
filter on geometry element
Monitor Number of tracks
UserTrackingAction
Monitor timing
Filter on particle type
UserEventAction
USerRunAction
User
produce
TXT log
Produce histograms
produce ntuples
Powered By Visual Paradigm Community Edition


Application
G4Run
G4MultiRunAction
G4MonitoringRunAction
G4Timer – Run
G4Event
G4MultiEventAction
G4MonitorinEventAction
G4Counter – Event
G4Timer – Event
Tracking
G4MultiTrackingAction
G4MonitoringTrackingAction
G4Counter – track
Filter
G4Timer – Track
Stepping
G4MultiSteppingAction
G4MonitoringSteppingAction
G4Counter – step
Filter
G4MonitoringLogger
1: BeamOn
1.1: BeginOfRunAction
1.1.1: BeginOfRunAction
1.1.1.1: Start
1.1.1.2: BeginOfEventAction
1.2: BeginOfEventAction
2: ProcessOneEvent
2.1: BeginOfEventAction
2.1.1: BeginOfEventAction
2.1.1.1: Process
2.1.1.1.1: Start
2.1.1.2: BeginOfEventAction
2.2: BeginOfEventAction
3: ProcessOneTrack
3.1: PreUserTrackingAction
3.1.1: PreUserTrackingAction
3.1.1.1: Process
3.1.1.1.1: Accept
3.1.1.1.1.1: NextFilter
3.1.1.1.1.2: Accept
3.1.1.1.2: Start
3.1.1.1.3: Process
3.1.2: PreUserTrackingAction
3.2: PreUserTrackingAction
4: ProcessOneStep
4.1: UserSteppingAction
4.1.1: UserSteppingAction
4.1.1.1: Process
4.1.1.1.1: Accept
4.1.1.1.1.1: NextFilter
4.1.1.1.1.2: Accept
4.1.1.2: Process
4.1.2: UserSteppingAction
4.2: UserSteppingAction
4.3: ProcessOneStep
5: PostUserTrackingAction
5.1: PostUserTrackingAction
5.1.1: StopTimer
5.1.1.1: Stop
5.1.2: DumpAndReset
5.1.2.1: log
5.1.3:
5.1.4:
6:
7: EndOfEventAction
7.1: EndOfEventAction
7.1.1: StopTimer
7.1.1.1: Stop
7.1.2: DumpAndReset
7.1.2.1: log
7.1.2.2:
7.2:
7.3:
8:
9: EndOfRunAction
9.1: EndOfRunAction
9.1.1: StopAndGetTime
9.1.2: StopAndGetTime
9.1.3: GetCount
9.1.4: GetCount
9.1.5: Fill
9.1.6: EndOfRunAction
9.1.7: EndOfRunAction
10: BeamOn
Powered By Visual Paradigm Community Edition


m_nextFilter
0..1
<<Interface>>
G4VLoggingFilter
#m_accept : bool
+Check()
+Process() : bool
+SetNextFilter(f : G4VLoggingFilter *)
0..*
D
0..1
G4TLoggingFilter
#m_data : D
+Accept(data : D) : bool
+Set(data : D)
<<bind>>
<G4Track*>
<<bind>>
<G4Step*>
G4TLoggingFilter ...
G4TLoggingFilter ...
G4LoggingFilterTrack
+Check()
G4LoggingFilterStep
+Check()
0..1
0..1
0..1
0..1
G4LoggingParticleFilter
-m_particle : G4ParticleDefinition*
+Matches(part : G4ParticleDefinition *) : bool
G4LoggingEnergyFilter
-m_min : double = 0
-m_max : double = DBL_MAX
+Matches(ene : double) : bool
0..1
0..1
G4LoggingLogicalVolumeFilter
-m_logVolumeName : std::string
+Matches(lv : G4LogicalVolume *) : bool
<<Interface>>
G4VLogging
#m_counter : G4int = 0
+Increment()
+Dump()
+AddTimer()
+GetTimer() : G4Timer &
+GetCounter() : G4int
+HasTimer() : G4bool
+Clone() : G4VLogging *
+Reset()
0..1
m_timer
0..*
G4Timer
m_filter
D
G4TLogging
-m_data : D
+Process(data : D) : bool
+Dump()
<<bind>>
<G4Run*>
<<bind>>
<G4Event*>
<<bind>>
<G4Track*>
<<bind>>
<G4Step*>
G4LoggingRun
G4LoggingEvent
G4LoggingTrack
G4LoggingStep
std::vector ...
std::vector ...
std::vector ...
std::vector ...
G4UserTrackin...
G4UserSteppin...
G4UserRunAction
G4UserEventActi...
G4MonitorTrackingAction
+PreUserTrackingAction(trk : G4Track *)
+PostUserTrackingAction(trk : G4Track *)
+DumpAndReset(evtID : int)
G4MonitorSteppingAction
+UserSteppingAction(stp : G4Step *)
+DumpAndReset()
0..1
0..1
0..1
0..1
G4MonitorEventAction
+BeginOfEventAction(evt : G4Event *)
+EndOfEventAction(evt : G4Event *)
+DumpAndReset(runID : int)
G4MonitorRunAction
+BeginOfRunAction(run : G4Run *)
+EndOfRunAction(run : G4Run *)
+DumpAndReset()
+GenerateRun() : G4Run *
0..1
0..1
0..1
m_io
<<Interface>>
G4VLoggingOutput
-prefix : std::string = "[LOG]"
+log(tag : std::string &, msg : std::string) : bool
+SetPrefix(str : std::string &)
+log(tag : std::string &, value : T) : bool
+log(tag : std::string &, time : G4Timer &) : bool
G4LoggingOutput
+log(tag : std::string &, msg : std::string) : bool
+log(tag : std::string &, value : T) : bool
+log(tag : std::string &, time : G4Timer &) : bool
G4LoggingOutputNtuple
+AddColumn(name : std::string)
G4LoggingOutputHisto
+BookHisto(title : std::string, nbins : int, xmin : double, xmax : double)
<<use>>
<<use>>
<<use>>
create
<<use>>
create
<<use>>
create
m_out
G4UserRunAction
g4analysis
G4LoggingOutputRunAction
+BeginOfRunAction(run : G4Run *)
+EndOfRunAction(run : G4Run *)
1
<<Enum>>
G4LoggingOutputType
<<use>>
G4MonitorManager
+SetOutputType(type : G4LoggingOutputType, filename : std::string & = "")
+Initialize()
+AddStepMonitor(mon : G4LoggingStep *)
+CreateStepMonitor(name : std::string & = "Step") : G4VLogging *
+AddParticleFilter(pdef : G4ParticleDefinition *, mon : G4LoggingStep *, parent : G4VLoggingFilter * = nullptr) : G4VLogging *
+AddLogicalVolumeFilter(lvName : std::string &, mon : G4VLogging *, parent : G4VLoggingFilter * = nullptr) : G4VLogging *
+AddEnergyFilter(min : G4double, mon : G4LoggingStep *, parent : G4VLoggingFilter * = nullptr, max : G4double = DBL_MAX) : G4VLogging *
+AddTrackMonitor(mon : G4LoggingTrack *)
+CreateTrackMonitor(name : std::string & = "Track") : G4VLogging *
+AddParticleFilter(pdef : G4ParticleDefinition *, mon : G4LoggingTrack *, parent : G4VLoggingFilter * = nullptr) : G4VLogging *
+AddLogicalVolumeFilter(lvName : std::string &, mon : G4LoggingTrack *, parent : G4VLoggingFilter * = nullptr) : G4VLogging *
+AddEnergyFilter(min : double, mon : G4LoggingFilterTrack *, parent : G4VLoggingFilter * = nullptr, max : double = DBL_MAX) : G4VLogging *
+CreateEventMonitor(name : std::string *) : G4VLogging *
+CreateRunMonitor(name : std::string &) : G4VLogging *
+AddEventMonitor(mon : G4MonitorEventAction *)
+AddRunMonitor(mon : G4MonitorRunAction *)
G4MonitorMessenger
1
Powered By Visual Paradigm Community Edition