# Proposed interface changes to `NuRandomService`

Kyle J. Knoepfel

LArSoft Coordination Meeting

23 October 2018

# Upgrading LArSoft to *art* 3

- Moving LArSoft to support multi-threading is a significant effort that is underway
  - Serial event-processing is *art*'s default behavior—getting here is the first step.
  - You must opt-in to multi-threaded execution.

- There is guidance for how to do this:
  - https://cdcvs.fnal.gov/redmine/projects/art/wiki/Upgrading_to_art_3

- Lynn and I have been upgrading the LArSoft repositories to support *art* 3.
  - Exposed some breaking changes I was not aware of
    - I am in the process of updating the *art* breaking-changes page
  - Exposed suboptimal practices (e.g.):
    - Lot of calls to `RandomNumberGenerator::getEngine(…)`, which will be deprecated in *art* 3.02 and removed in *art* 3.03.

# NuRandomService interface change

- NuRandomService is widely used in the LArSoft repositories. Its createEngine provides a layer on top of *art*'s createEngine interface.
  - The only way to interact with the random-number engine is to call RandomNumberGenerator::getEngine
  - Such a call is expensive, exposes multi-threading details to the user, and is unnecessary.
  - In *art* 3.02 it will be deprecated; in *art* 3.03 it will be removed.

- Like *art*'s createEngine interface, NuRandomService::createEngine will return a reference to the *art*-managed engine.

```
- long seed = ServiceHandle<NuRandomService>{}->createEngine(...);
+ CLHEP::HepRandomEngine& engine = ServiceHandle<NuRandomService>{}->createEngine(...);
```

- No code breaks in LArSoft; not sure about experiment repositories.

# NuRandomService interface change

## Before

```
class MyProducer {
public:

  MyProducer(ParameterSet const& pset)
  {
    ServiceHandle<NuRandomService>{}
      ->createEngine(...);
  }

  void produce(art::Event& e) override
  {
    auto& engine =
      ServiceHandle<RandomNumberGenerator>{}
        ->getEngine(...);
    CLHEP::RandFlat flatDist{engine};
    flatDist.fire(...);
  }
};
```

# NuRandomService interface change

## Before

```
class MyProducer {
public:

  MyProducer(ParameterSet const& pset)
  {
    ServiceHandle<NuRandomService>{}
      ->createEngine(...);
  }

  void produce(art::Event& e) override
  {
    auto& engine =
      ServiceHandle<RandomNumberGenerator>{}
        ->getEngine(...);
    CLHEP::RandFlat flatDist{engine};
    flatDist.fire(...);
  }
};
```

***Expensive operations:***

- *`ServiceHandle` created for each event*
- *`getEngine` called on each event*
- *`RandFlat` distribution created for each event*

# NuRandomService interface change

## Before

```
class MyProducer {
public:

  MyProducer(ParameterSet const& pset)
  {
    ServiceHandle<NuRandomService>{}
      ->createEngine(...);
  }

  void produce(art::Event& e) override
  {
    auto& engine =
      ServiceHandle<RandomNumberGenerator>{}
        ->getEngine(...);
    CLHEP::RandFlat flatDist{engine};
    flatDist.fire(...);
  }
};
```

## After

```
class MyProducer {
  CLHEP::RandFlat flatDist_;

public:

  MyProducer(ParameterSet const& pset)
    : flatDist_{ServiceHandle<NuRandomService>{}
                  ->createEngine(...)}
  {}

  void produce(art::Event& e) override
  {
    flatDist_.fire(...);
  }
};
```

# NuRandomService interface change

## Before

```
class MyProducer {
public:

  MyProducer(ParameterSet const& pset)
  {
    ServiceHandle<NuRandomService>{}
      ->createE
  }

  void produce(art::Event& e) override
  {
    auto& engine =
      ServiceHandle<RandomNumberGenerator>{}
        ->getEngine(...);
    CLHEP::RandFlat flatDist{engine};
    flatDist.fire(...);
  }
};
```

*createEngine returns art-owned reference to engine; no need to directly interact with it*

## After

```
class MyProducer {
  CLHEP::RandFlat flatDist_;

public:

  MyProducer(ParameterSet const& pset)
    : flatDist_{ServiceHandle<NuRandomService>{}
                  ->createEngine(...)}
  {}

  void produce(art::Event& e) override
  {
    flatDist_.fire(...);
  }
};
```

# Additionally…

- There is no notion of “current module” in MT code:
  - I have removed `getCurrentSeed` and `getGlobalCurrentSeed` (no one calls them)

- `NuRandomService` internally uses `RandomNumberGenerator::getEngine`.
  - I am working on ways to remove this so that it is no longer necessary
  - Expect more interface changes in the future

# The proposal

- Change the createEngine interface to return a reference to the *art*-managed random-number engine:

```
- long seed = ServiceHandle<NuRandomService>{}->createEngine(...);
+ CLHEP::HepRandomEngine& engine = ServiceHandle<NuRandomService>{}->createEngine(...);
```

- Remove `getCurrentSeed` and `getGlobalCurrentSeed`

- Feature branches available:
  - `nutools:feature/knoepfel_createEngine_interface`
  - `larsim:feature/knoepfel_createEngine_interface`
  - `larreco:feature/knoepfel_createEngine_interface`
  - `larana:feature/knoepfel_createEngine_interface`