



## *art news*

Kyle J. Knoepfel  
*art* stakeholders meeting  
19 January 2017



## art 2.06

- *art* 2.06 is nearing completion:
  - 6.08/04 is has been released with the bug fixes we needed.
  - Extended `art::Assns` behavior is being implemented in gallery.
  - Will include GCC 6.3 (e14).
  - Significant improvements to ownership semantics (under the covers).

# Removal of on-demand support (for now)

- *art* currently supports on-demand production of Event-level products.
  - i.e. products are created whenever one attempts to retrieve them via (e.g.) `Event::getValidHandle<ProductType>(tag)`.
  - Various weaknesses exist with the current *art* implementation of this feature.
  - Retaining its implementation is likely to make moving to multi-threading more difficult.
  - **Since no experiment appears to be using it (based on our searches in experiment repositories), the *art* team would like to remove the feature for now.**
  - This does not preclude *art* from incorporating on-demand production in the future. (CMSSW uses on-demand reconstruction extensively.)

## Changes to `art::Source<T>` behavior

- For jobs that use an input source that is an instantiation of `art::Source<T>`, `art`'s event loop is governed by the user's `readNext` function.

```
bool readNext(art::RunPrincipal const* const inR,  
              art::SubRunPrincipal const* const inSR,  
              art::RunPrincipal*& outR,  
              art::SubRunPrincipal*& outSR,  
              art::EventPrincipal*& outE);
```

## Changes to `art::Source<T>` behavior

- For jobs that use an input source that is an instantiation of `art::Source<T>`, `art`'s event loop is governed by the user's `readNext` function.

```
bool readNext(art::RunPrincipal const* const inR,  
              art::SubRunPrincipal const* const inSR,  
              art::RunPrincipal*& outR,
```

Pointers to cached Run and SubRun principals.  
Will be null for the first call to `readNext`.

## Changes to `art::Source<T>` behavior

- For jobs that use an input source that is an instantiation of `art::Source<T>`, `art`'s event loop is governed by the user's `readNext` function.

```
bool readNext(art::RunPrincipal const* const inR,  
             art::SubRunPrincipal const* const inSR,  
             art::RunPrincipal*& outR,  
             art::SubRunPrincipal*& outSR,  
             art::EventPrincipal*& outE);
```

Pointers (initialized to `nullptr`), that the user can set to drive `art` state transitions. Usually requires using `inR` and `inSR` for creating new `SubRun` and `Event` principals.

# Changes to `art::Source<T>` behavior

- For jobs that use an input source that is an instantiation of `art::Source<T>`, *art*'s event loop is governed by the user's `readNext` function.

```
bool readNext(art::RunPrincipal const* const inR,  
              art::SubRunPrincipal const* const inSR, ...);
```

- In current *art*, the `inR` and `inSR` principals are cached across input file boundaries. Users are thus not required to set `outR` and `outSR` before setting `outE` when a new input file is opened.
- Such caching can be problematic, as was been encountered in earlier versions of *art* with `RootInput`. With *art* 2.01, such caching was disabled for `RootInput`, and the system was revamped to allow a consistent interpretation of `Run` and `SubRun` objects and products.
- The caching was not disabled for `art::Source<T>`, leading to an inconsistent implementation, and potential problems for `art::Source<T>` users.

# Changes to `art::Source<T>` behavior

- For jobs that use an input source that is an instantiation of `art::Source<T>`, *art*'s event loop is governed by the user's `readNext` function.

```
bool readNext(art::RunPrincipal const* const inR,  
              art::SubRunPrincipal const* const inSR, ...);
```

- For *art* 2.06, whenever a new input file is opened, the `inR` and `inSR` principal pointers will be reset to `nullptr`.
- The user will need to set `outR` (and possibly `outSR`) when *any* input file is opened, not just the first input file.
- *art* is **not** imposing a file-format on users. However, *art* will expect users to prepare the appropriate principals so that the *art* state-machine transitions are correctly respected.

# Allowed transitions between levels

