Agenda: 2 separate talks!

Don’t mangle together what does
not belong together

1. **Schema Evolution Tests**
2. **Dictionaries on Demand**

Schema Evolution Tests

- POOL/ROOT Schema evolution
- Motivation
- Planned tests

Goals

- **Schema evolution**
  - Proof that POOL inherits the ROOT schema evolution
    No more – no less
    No extras
- **Dictionary on demand**
  - Late loading of dictionaries

Schema Evolution

- Root provides schema evolution
- POOL inherits the ROOT schema evolution
  - class TStreamerInfo
  - No extra effort inside POOL necessary
- But it has to be tested…
- Establish test matrix
- Cross check with native root
  ie. rootcint generated dictionaries

Schema Evolution Test Matrix

	Handled by ROOT	Handled by POOL
Change class type		
change namespace	?	?
to related type	?	?
to unrelated type	?	?
Add new data member	?	?
Remove data member	?	?
Rename data member	?	?
Change data member type		
change namespace	?	?
to related type	?	?
to unrelated type	?	?
change primitive type	?	?

Timescale

- After the move to ROOT 4.02
- Results (and possible fixes) around Easter

On Demand Dictionary Loading

- Usage of dictionaries
- Circumstances/boundary conditions
- Technical details
- Required changes

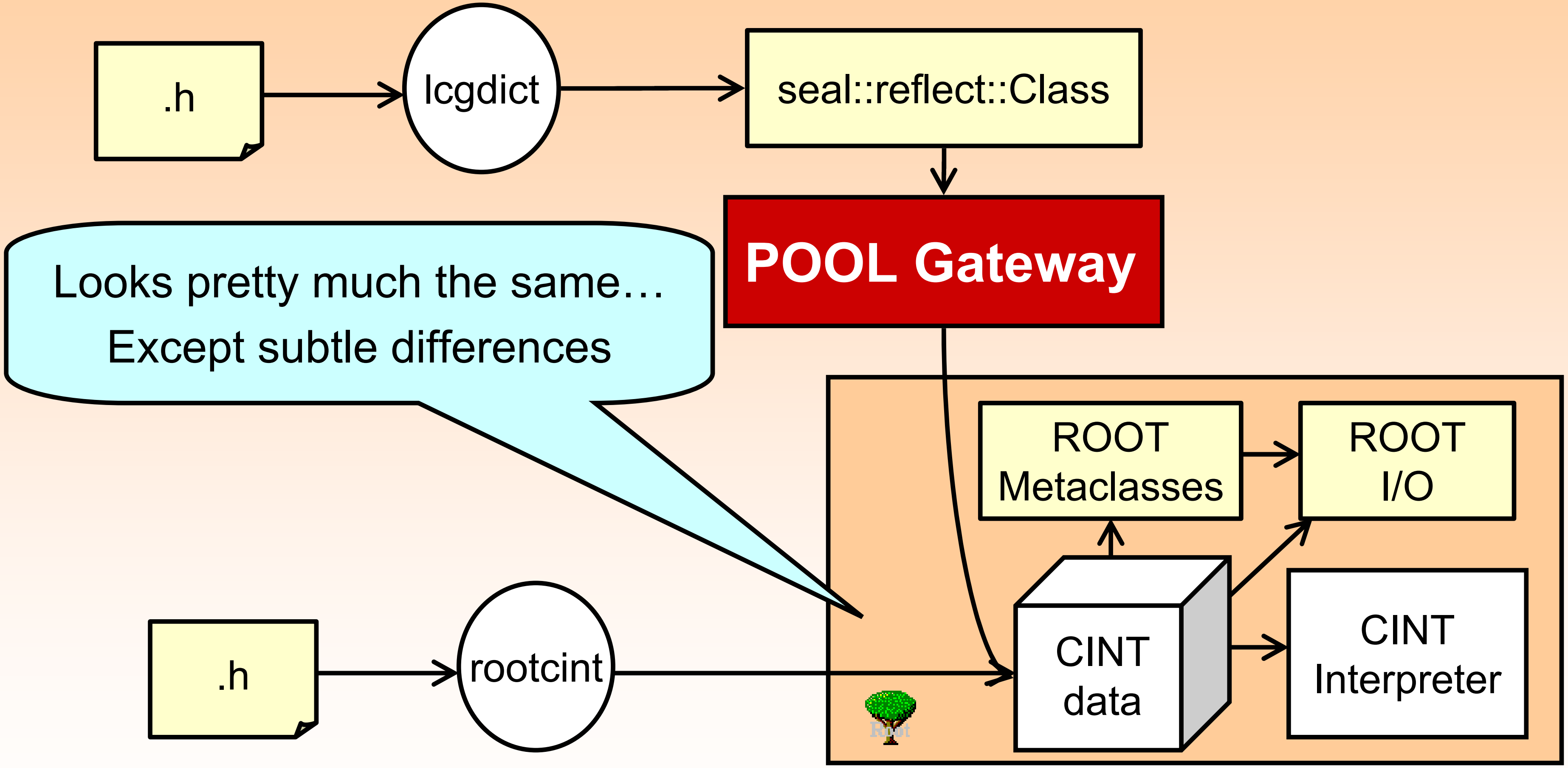
Dictionaries on Demand

- **Dictionaries are necessary**
  - At write time for the classes to be written
  - At read time when they are requested by ROOT

- **Any class dictionary must be *complete***
  - The class itself
  - Any base class
  - Any aggregate/association

Let’s analyze the current situation

POOL and ROOT Reflection



Looks pretty much the same…
Except subtle differences

What are the Differences ?

- **rootcint provides a callback when a class comes online / offline**
  - This allows to convert “fake” classes on the fly to “real” classes
- **SEAL reflection *currently* does not**
  - Will be part of Reflex (evolution of SEAL dictionary)
- **ROOT and SEAL class names are the same …*most* of the time**
  - e.g. ROOT/CINT does not use the “std” namespace (remember gcc 2.95 was the same ?)

Consequences (1)

- POOL registers a “TClassGenerator” module to ROOT,
  - *retrieves load requests and must satisfy them*
  - *Requests are expressed in the names known to ROOT*
  - *There is no second notification*
- ROOT / CINT meta-classes can only created by the POOL gateway is a SEAL reflection class is available
  - The requested dictionary module must be loaded

Consequences (2)

- ROOT wants to know the data member offsets to generate TStreamerInfo(s) and calls
  - *"ShowMembers", which requires an object instance*

For any dictionary translation we need

- A valid LCG dictionary
- A valid object instance

Possible Solutions

- **Use the SEAL dictionary service**
  - Would work well for reading and writing as long as
  - As long as there are **no abstract classes** and an object instance can easily be created
  - The **std namespace is not in use**
  - Based on capability information (I guess)
    No library loading, but no inheritance information either

- **Otherwise:**
  Need to scan all existing classes and look for
  - ROOT – SEAL name translation match
  - Concrete meta-classes of abstract types

Hence, the Problems are...

- ➢ Need to blindly scan for classes
- ➢ At this moment the dictionary service is no longer helpful
  - ➢ Likely to end up with even more dictionaries in memory

Any Feasible Solution

1. **Requires that the POOL gateway keeps track of all dictionaries which are**
   - Currently “on hold” requested by ROOT
     - “on hold” = described by a “fake class”
     - This is done by ROOT using TClassGenerator
   - Currently “online” in the seal reflection
     - Need to migrate to Reflex
2. **Requires POOL to be notified**
   - When ROOT uses a metaclass the first time (while reading), not or not only when opening the file

Any Feasible Solution (2)

3. **Requires that the bloody type names become standardized**
    - Remember: Everything is bound using strings
    - And this is a huge mess
        - There is no standard and no agreement whatsoever !
        - Std or not …
        - "using" or not
        - short signed, short signed int, signed short int
they all come along - name them !!
    - CINT, gccxml, cxxabi and VC++ I know to be different
    - Probably every compiler has it's own opinion

Timescale

- **After the move to ROOT 4.02**
- **After the move to Reflex**
  - All of POOL needs to move
    it will enter at the top level interfaces
- **Hopefully after a standardization of type names**
  - If there is any hope at all
  - But I stll strongly believe there should be a unambiguous uniquely reversible transformation between the **type name** and the **typeid**
  - **It is feasible, but it needs collaboration between SEAL – ROOT – POOL**
- **No timescale**

Conclusions

- **The ROOT schema evolution mechanisms must be tested**
  - After the move to ROOT 4.02
  - Hopefully results will be present until Easter
  - The testing could also be a project for outside contributors

- **Dictionaries on Demand**
  - Require the next generation reflection interface
  - Preferably receive a callback from ROOT on the first usage of a meta-class
  - Require some common naming conventions