CINTEX: Filling CINT structures from Reflex

Applications Area Meeting

19 January 2005

P. Mato / CERN

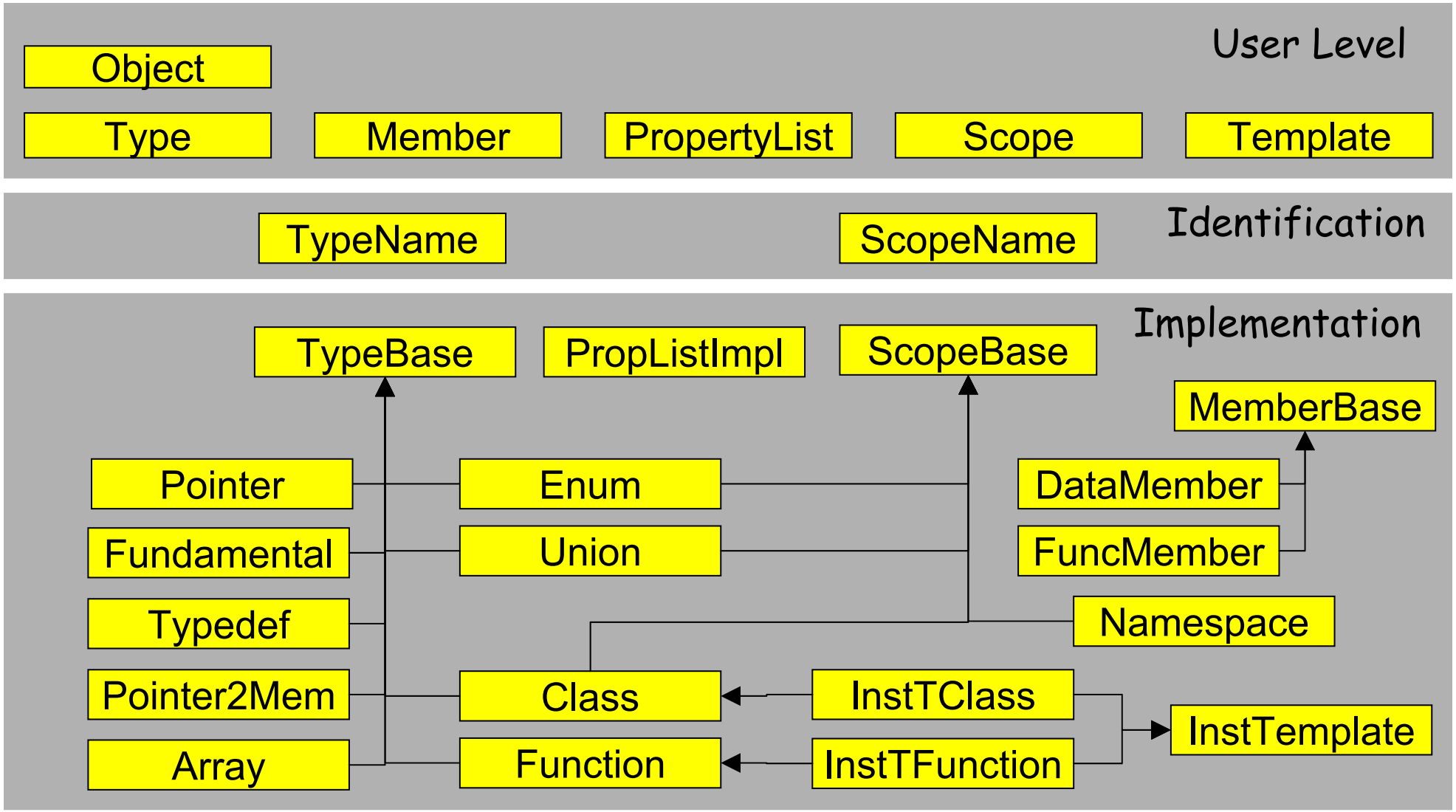
Contents

- Update on Reflex
- Current Reflex/ROOT strategy
- What is Cintex
- Current status
- Next steps
- Summary

Reflex: Main Goals

- Reflection is the ability of a language to introspect it's own structure at runtime and interact with it in a generic way
- Enhance C++ with reflection capabilities
  - Non intrusive, automated
- Close to the C++ ISO 14882 standard
- Lightweight and standalone system
  - Minimal dependencies
- Small memory footprint
- Multi platform
  - Linux, Windows, Mac OSX, …

Reflex Classes



Example: Introspecting types

```
// Get type by its name
Type cl = Type::byName("Particle");

// If class print all data members
if ( cl.isClass() ) {
  for ( size_t d = 0; d < cl.dataMemberCount(); d++ ) {
    Member dm = cl.dataMember(d);
    cout << dm.type().name(SCOPED) << " " << dm.name() <<";";
    // output comment line if exists
    if ( dm.propertyList().hasKey("comment") ) {
      cout <<"//"<< m.propertyList().propertyAsString("comment");
    }
    cout << endl;
  }
}
```

Example: Generic Interaction

```
// Get a type by its name
Type cl = Type::byName("Particle");

// Instantiate an instance
Object obj = cl.construct();

// Call a method
vector<void*> args;
Object ret = obj.invoke("function", args );

// Alternatively
for ( size_t f = 0; f < cl.functionMemberCount(); f++ ) {
  if (cl.functionMember(d).name() == "function") {
    ret = cl.functionMember(d).invoke(obj, args);
  }
}
// Delete the instance
cl.destruct(obj);
```

Reflex: Status

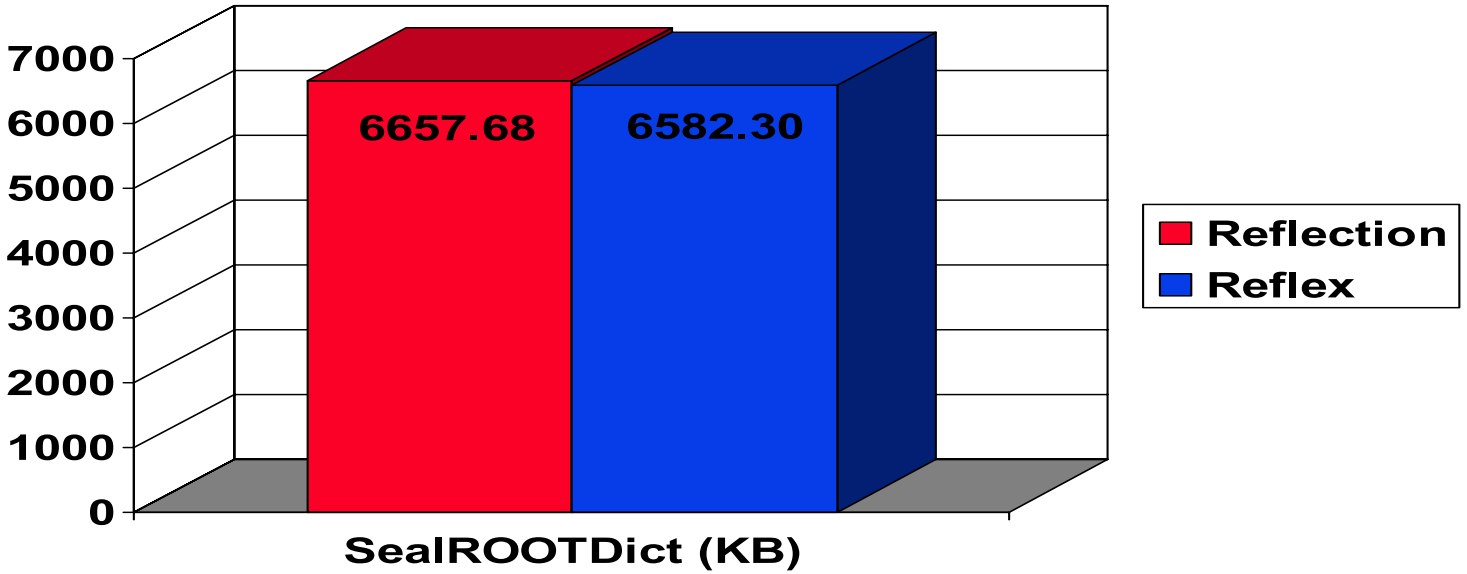
- Most of the functionality is implemented
  - Still missing full template support and other bits
- Dictionaries are generated with the "lcgdict" command (GCC_XML)

  --reflex option

  extended functionality: free functions, typedefs, etc.
- (Pre) Released as part of SEAL 1.6.0
- Performance and dictionary sizes very reasonable

Dictionary Library Sizes

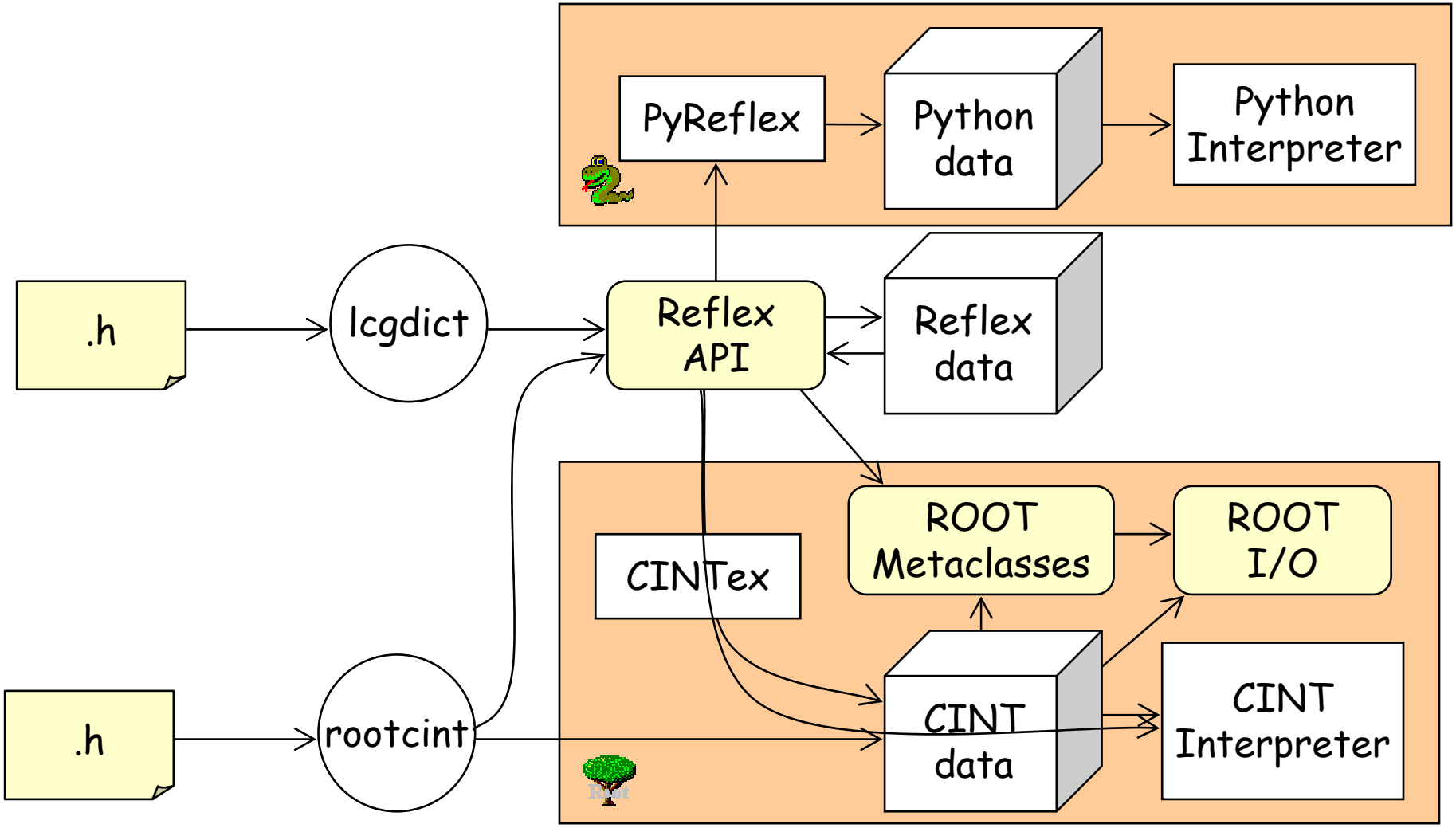
- SEAL ROOT Dictionary: 405 classes



Reflex/ROOT – (December Plans)

1. Fill CINT data structures (data and methods) from Reflex on demand.
   - This is needed to allow interactive work using CINT (ROOT) for classes for which only the Reflex dictionary exists. The code for this already exists in POOL for the LCG/CINT dictionary gateway.
   - Estimated to 3-4 months
2. Re-implement the ROOT metaclasses (TClass, TMethod, etc.) on top of Reflex
   - Estimated to 2 months
3. Adaptation of the CINT interpreter to run on top of Reflex directly is foreseen in principle but detailed planning will only be done after tasks 1 and 2 are completed.

Reflex and ROOT



Cintex: Driving Use Case

- Be able to write "simple" analysis macros (CINT) accessing an existing file produced by POOL without the need of loading POOL+SEAL+Exp-Framework+…
  - Cintex and Event classes (Reflex) dictionaries need to be loaded
  - Event class methods would be available
- Some Caveats
  - Strongly dependent on the "Event Model" design quality
  - POOL specialized streamers not available
  - POOL references not available (more work needed to understand)

Cintex: Current Functionality

- At loading time, Cintex registers itself to Reflex to get callback when a new classes are defined in Reflex
- For each class
  - Calls the appropriate CINT functions to create the class with their data and function members, and inheritance tree.
  - Creates the necessary namespaces and "forward declares" other types on-the-fly
- Provides a set of "generic" adapters for the "stub" functions between CINT and Reflex
- Current functionality quite complete
- Most of the knowledge (and code) taken from the POOL RootStorageSvc (Markus Frank)

Example: Interactive session

```
gSystem->Load("lcg_Cintex");    // Load Cintex
gSystem->Load("SealCLHEPDict"); // Load any Reflex dictionary

using namespace CLHEP;
Hep3Vector v1(10.,20.,30.);
Hep3Vector v2(v1);
cout << v2.r() << endl;

RanluxEngine r;
RandFlat f(r);
RandGauss g(r,0,1);
TH1F hf("hf","flat distribution",100,0,1);
TH1F hg("hg","gauss distribution",100,-5,5);
for (int i = 0; i < 10000; i ++) {
  hf.Fill(f.fire());
  hg.Fill(g.fire());
}
```

Example: Simple I/O example

```
gSystem->Load("lcg_Cintex");    // Load Cintex
gSystem->Load("SealCLHEPDict"); // Load any Reflex dictionary

CLHEP::Hep3Vector v0;
CLHEP::Hep3Vector v1(22,1,1);

TFile fo("data.root","RECREATE");
fo.WriteObjectAny(&v0, "CLHEP::Hep3Vector","my_v0");
fo.WriteObjectAny(&v1, "CLHEP::Hep3Vector","my_v1");
fo.Close();

TFile fi("data.root");
CLHEP::Hep3Vector* vp;
vp = (CLHEP::Hep3Vector*)fi.FindObjectAny("my_v1");
cout << " x = " << vp->x()
     << " y = " << vp->y()
     << " z = " << vp->z() << endl;
fi.Close();
```

Cintex: Current Limitations

- CINT optimization switched off
  - In optimize mode the single stub function adapter can not obtain the "context"
  - `gROOT->ProcessLine(".O 0");`
- Virtual inheritance
  - Not yet there. No major problems foreseen.
- I/O problems
  - Writing seems to work always (spurious messages)
  - Sometimes data misalignment problems on reading
  - ➔ The main use case in danger!

Example: Accessing POOL files

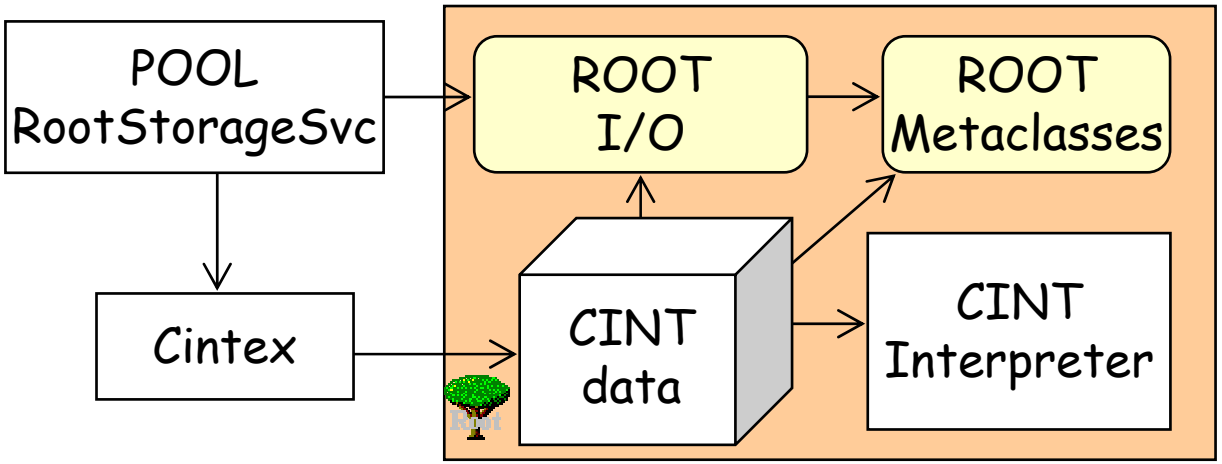
```
gROOT->ProcessLine(".O 0");
gSystem->Load("liblcg_Cintex");
gSystem->Load("libEventModelDict");

using namespace pool_tutorial;

TFile f("pool_tutorial_1.root");
Hit* h = 0;
TTree* tree = (TTree*)f.Get("hits");
tree->SetBranchAddress("Hit", &h);

int n = tree->GetEntries();
for ( int i = 0; i < n; i++ ) {
  tree->GetEntry(i);
  cout << "Hit " << i << ": " << h->x() << " " << h->y()
       << " " << h->z() << " " << h->value() << endl;
}
f.Close();
```

Unfortunately does not work today

Next steps

- (Pre) Released as part of SEAL 1.6.0
  - Requires new Reflex dictionaries + ROOT 4
- Complete the basic functionality
  - Virtual inheritance, I/O read misalignments, templates, variables, etc.
  - Some optimization still possible
- Try with a "real" use case
  - E.g. An experiment POOL file
  - Looking for a contact person from ATLAS and CMS
- Defined the roles of Cintex (CINT part) and the RootStorageSvc in POOL (I/O part)

Cintex and RootStorageSvc



- RootStorageSvc gets simple
- Separation of concerns
  - POOL should use Cintex as the Reflex/CINT gateway
  - POOL should interact with ROOT I/O for I/O related issues
    - » Special streamers, object references, optimization, etc.

Summary

- Reflex is getting better and functionally completed
  - Reduced the number of "user level" classes
  - Ready to be integrated in PyReflex, POOL, etc.
- Cintex provides the possibility to use CINT (ROOT) for any class with a Reflex dictionary
- The first task in the Reflex/ROOT convergence is "almost" completed
  - Requires validation from experiments (ATLAS and CMS)
  - Requesting contact person to try Cintex in their environment
- Next steps
  - Adaptation of POOL (RootStorageMgr) to Reflex and Cintex
  - Start implementing second task of Reflex/ROOT convergence plan