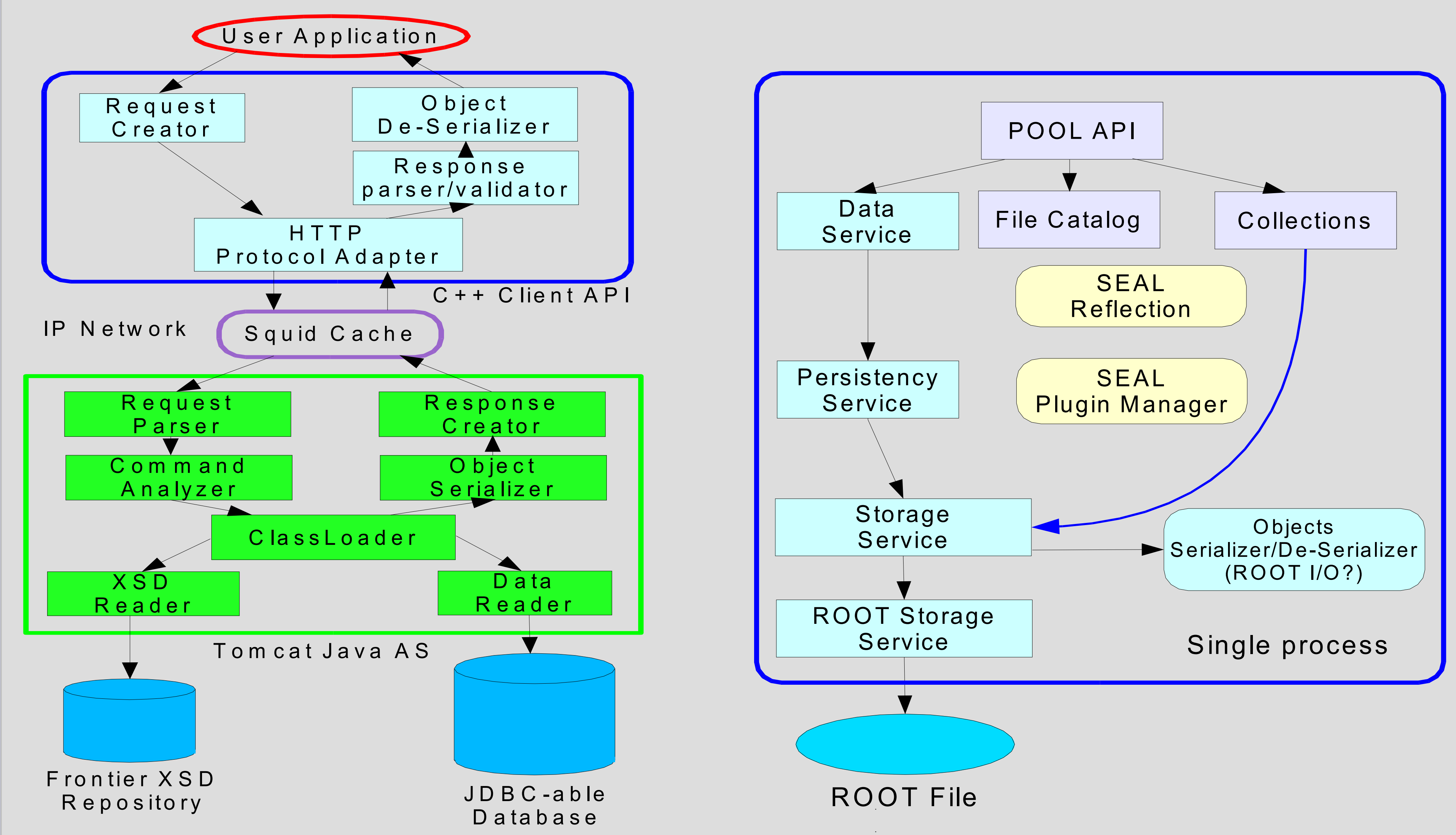
Frontier integration into LCG POOL

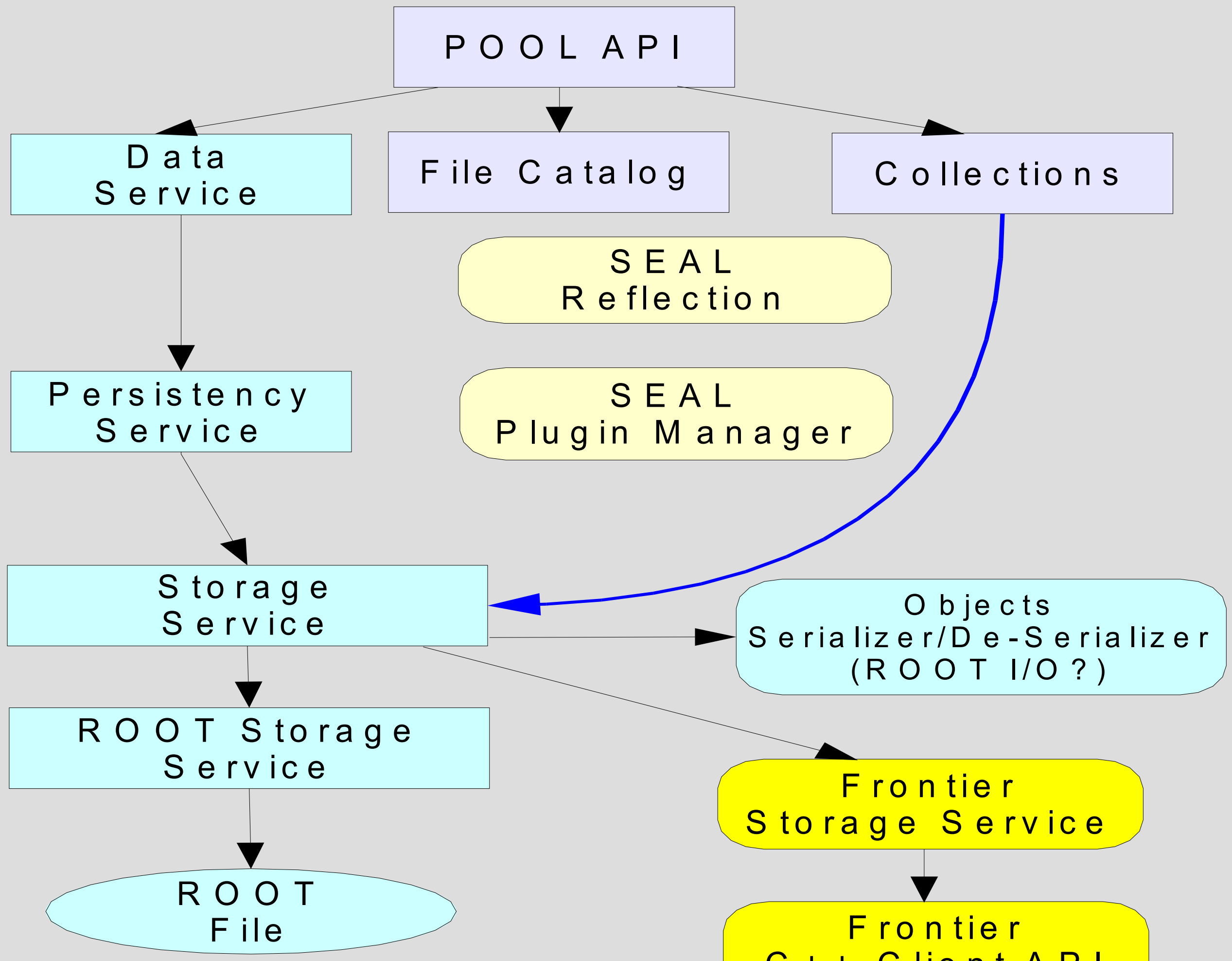
Frontier & POOL (Simplified)



Goal: access CMS calibration data using POOL via Frontier

- Create set of classes representing CMS calibration data (5 classes so far)
- Create Frontier XSD for each calibration class
- Generate SEAL reflection dictionaries for these classes
- Create plugin for POOL to request data from Frontier and map it into objects of calibration classes – the major challenge.

FrontierStorageSvc



Approach #1: FrontierStorageSvc using Db*Imp

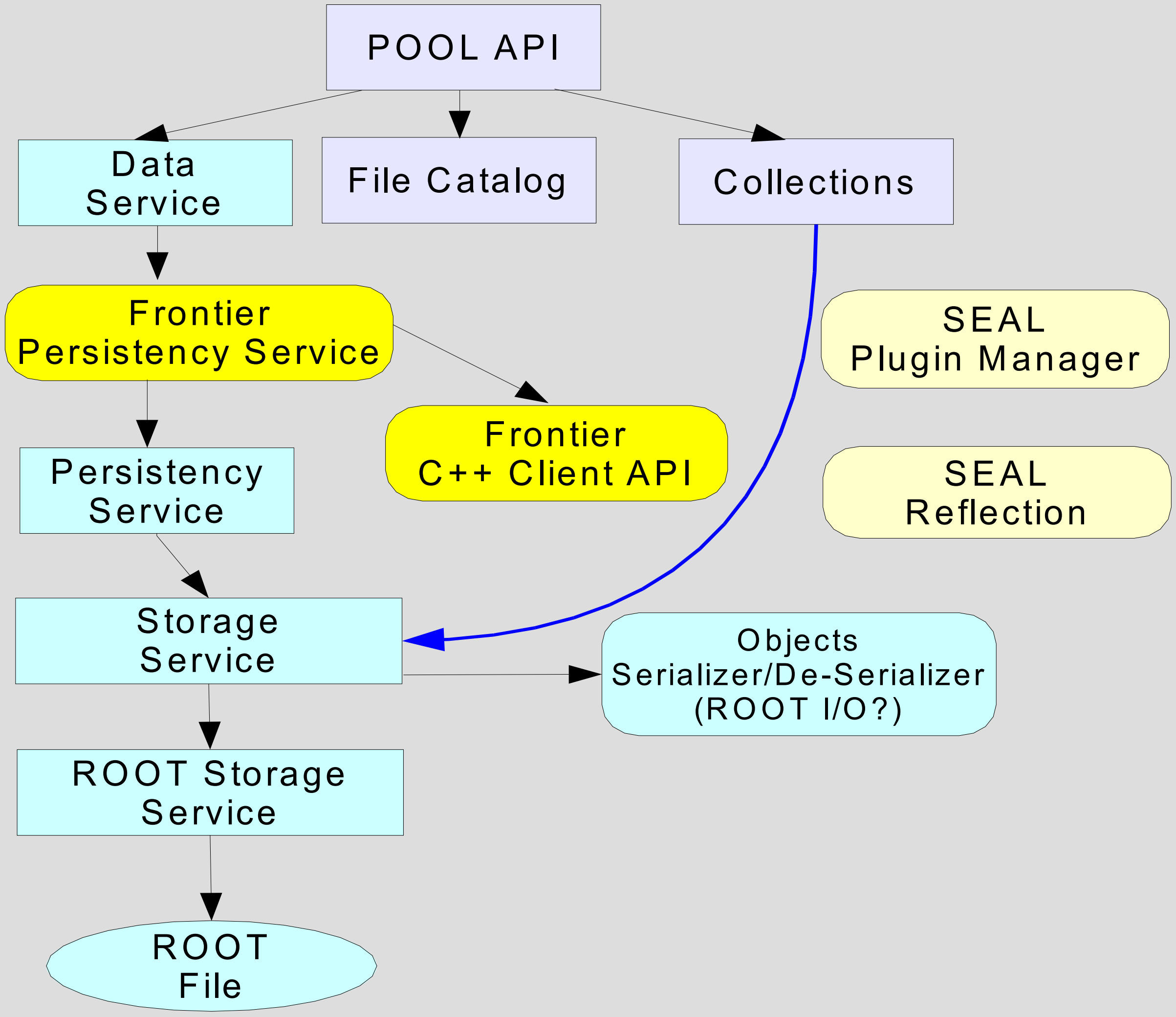
Module FrontierStorageSvc based on RootStorageSvc implementing Db*Imp-classes (OODatabaseImp, DbContainerImp, DbDomainImp, etc). These classes are part of framework from StorageSvc sub-project.

- This approach would provide seamless integration with POOL
- Problems:
  - The framework expects special information in three containers: ##Shapes, ##Links, ##Params. The info is set of strings which is generated when an object is written into storage.
  - The framework expects the object to be serialized in predefined way and stored as a single column (?)

Approach #2 – FrontierStorageSvc using IDb* interfaces

- Reimplementation of FrontierStorageSvc using IDb* interfaces from StorageSvc sub-project: IOODatabase, IDbDatabase, IDbContainer, etc.
- Problems:
  - IDb* interfaces bounded to the same Db*Imp framework from StorageSvc. For exmaple, IDbCursor uses DbContainer which call for ##Shapes, ##Links and ##Params containers. So this approach did not fix the problem.
  - Moking these containers is not straightforward and does not look as a good solution (compatibility problem, for example).
  - Does not solve “single column” problem.

FrontierPersistencySvc



Approach #3 - FrontierPersistencySvc

- IPersistencySvc can be used to control DataSvc context
- FrontierPersistencySvc is an implementation of IpersistencySvc.
- It actually implements only `readObject(Token &token, DataHandler handler).` All other requests are forwarded to default PersistencySvc implementation (which is obtained using factory).
- Sample application which reads ROOT stored object from ROOT collection and then reads GainErrorAllByTagTimei from Frontier works.
- Collection interface does not work for in this case. Does Collection make sense for Frontier?
- Hidden obstacles? E.g. `readObject()` is expected to change state of other modules?

POOL via Frontier – general object selection problem

- POOL uses Tokens to identify objects. Token consists of ClassID (GUID), database name, container name, technology ID (e.g. ROOT_Key or Frontier), type ID and object ID. The last is unique object identifier (?) and is a `pair<long,long>`.
- Frontier requests objects by server name, object name, object version, and 0 or more paires key_name:key_value. The same object could be potentially requested in multiple ways.
- POOL's Token does not provide ability for multiple key_name:key_value parameters. One of the approaches – use container name as encoded request string.

POOL via Frontier – serializer problem

- Frontier has it own platform-independend and network safe binary objects serializer and de-serializer
- POOL expects objects to be serialized in particular way (Shape?)
- Re-shaping Frontier objects for POOL would be inefficient, and hard to implement (because of ##Shapes, ##Links and ##Params containers the StorageSvc expects).

CMS calibration class example

```
class GainErrorAllByTagTimeiStruct
 {
  public:
  int eta;
  double phi;
  int depth;
  double value;
  double sigma;

  virtual ~GainErrorAllByTagTimeiStruct(){}
 };

class GainErrorAllByTagTimei
 {
  public:
  std::vector<GainErrorAllByTagTimeiStruct> data;

  static const char *getGUID(){return "4AF7DAD1-E51E-D911-829B-000D616B4939";}

  void append(void *ptr)
   {
    data.insert(data.end(),(GainErrorAllByTagTimeiStruct*)s);
   }

  virtual ~GainErrorAllByTagTimei(){}
 };
```

Sample application snaplet

```
pool::IFileCatalog      *catalog;
...
pool::FrontierPersistencySvc *ipsvc=new pool::FrontierPersistencySvc(*catalog);
pool::IDataSvc *context=pool::DataSvcFactory::instance(ipsvc);

seal::SharedLibrary::load(seal::SharedLibrary::libname("FrontierDict"));
seal::SharedLibrary::load(seal::SharedLibrary::libname("SealSTLDict"));

context->transaction().start(pool::ITransaction::READ);

pool::Guid guid(GainErrorAllByTagTimei::getGUID()); // GainErrorAllByTagTimei
pool::Token tok;
tok.setTechnology(pool::FRONTIER_StorageType.type());
tok.setDb("http://edge.fnal.gov:8000/FrontierCMS");
tok.setCont("tag='TAG'&date_from='08/01/2004'&date_to='09/01/2004'");
tok.setClassID(guid);

pool::Ref<GainErrorAllByTagTimei> ref(context,tok);

std::cout<<"eta="<<ref->data[0].eta<<'\n';
std::cout<<"eta="<<ref->data[1].eta<<'\n';
std::cout<<"eta="<<ref->data[99].eta<<'\n';
```