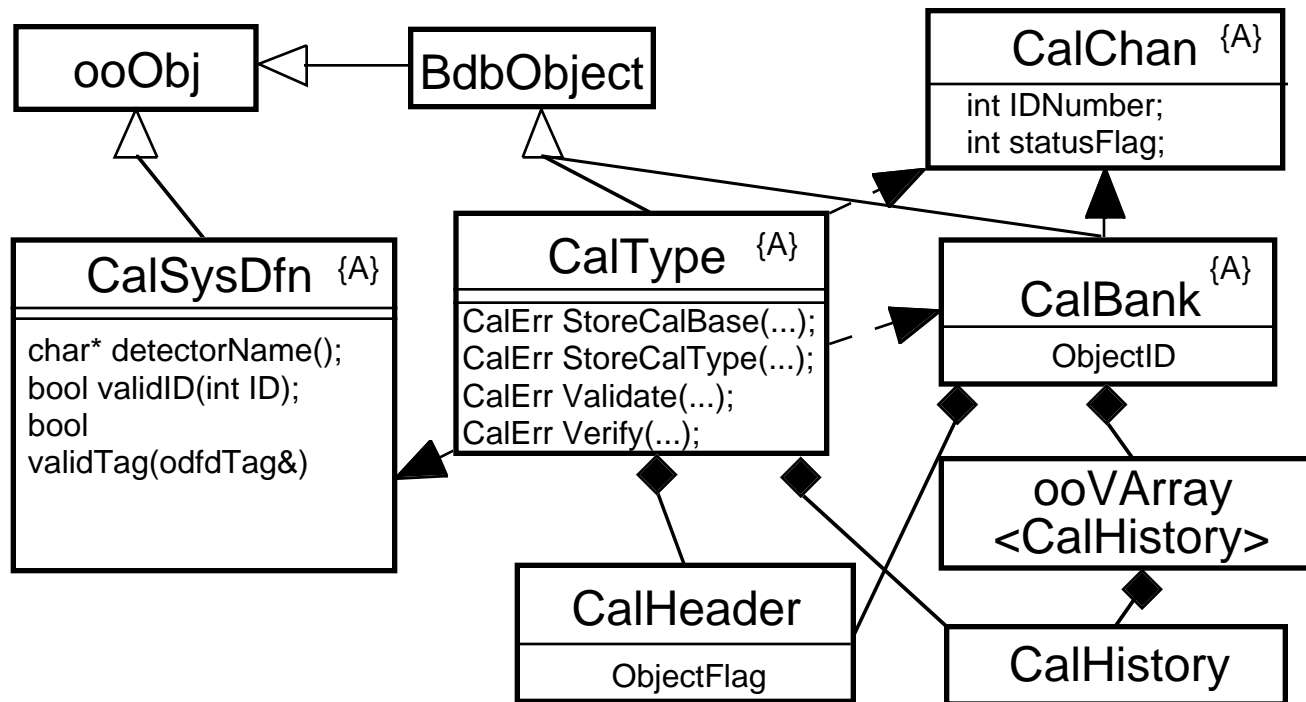


Conditions Database Applications

- ◆ Calibration
- ◆ Geometry + Alignment
- ◆ Offline Interface
 - ◆ Reading (XxxEnv + proxies)
 - ◆ Writing (BdbCondLoad)
- ◆ What's missing

Calibration use of Conditions DB

- ◆ Core calibration DB code in place since >1 year
 - ◆ Read/Write access provided through generic transient proxies
 - ◆ Write access limited by persistent 'gatekeeper' objects to Insure data integrity and quality
 - ◆ Provides a standard higher-level interface to calibration data across subsystems
- ◆ Most subsystems have experimented with this
 - ◆ Only Drc + Emc have implemented specialized subclasses (?)
- ◆ Dirc has functional examples using calib. data in Reco (T0)



Geometry and Alignment

- ◆ Standard offline interface to Geometry + Alignment
 - ◆ DetectorModel
- ◆ Nominal geometry in offline from conditions DB objects
 - ◆ All 5 subsystems provide this capability
 - ◆ Drc + Svt no longer support non-objy geometry
- ◆ Alignment corrections in reconstruction from conditions DB
 - ◆ Drc, Emc, + Svt support global alignments
 - ◆ Svt supports local (single wafer) alignments, Drc supports barbox alignment
 - ◆ Persistent classes use the same data primitive (DetAlignElem)
- ◆ Alignment output being written to the conditions DB
 - ◆ Dch + Svt global alignment modules

Conditions DB Offline Read Access

- ◆ Offline accesses conditions data through transient proxies
 - ◆ Shields end developers from knowing about Objectivity
 - ◆ Proxy mechanism allows 'automatic' following of time dependence
 - ◆ Examples (released code, functioning as part of reconstruction)
 - ◆ SvtDetectorProxy provides Svt geometry corrected for global and local alignment
 - ◆ DrcTDCToTimeProxy provides the factory which converts TDC counts to time (uses calibration)
 - ◆ These examples being used as templates for other systems
- ◆ Users access the proxies through their environment (XxxEnv)
 - ◆ Pre-objy anachronism, provides no functionality in ProxyDict, Objy world
 - ◆ Dependency collection point for conditions transients
 - ◆ Can (should) we migrate away from this interface?

Conditions DB Loading

- ◆ Conditions DB is loaded when user creates a test release
 - ◆ BdbCondLoad executable part of gmake database.import
 - ◆ Specific code (Modules) is provided by the subsystems
 - ◆ All subsystems have contributed a module to this executable
 - ◆ BdbCondLoad loads data expected by normal reco modules
 - ◆ Data ultimately comes from flat files or from the code
 - ◆ Easily modified
 - ◆ Can be personalized
 - ◆ Can load nominal + 'distorted' data
 - ◆ Allows studying alignment + offline calibration with common test vectors
 - ◆ Distorted data is separated from nominal in time (by subsystem)
- ◆ BdbCondLoad is a temporary measure
 - ◆ Avoids schema evolution problems
 - ◆ Avoids problems copying databases
 - ◆ No valuable data needs to be carried between offline releases (yet)

Outstanding Issues

- ◆ Write access to Cond. DB is too slow for online calibration
 - ◆ Writing a full calibration takes 10s of seconds
 - ◆ ~100 X slower than raw Objy write of simple objects
 - ◆ Problem may lie in Objy, Conditions DB interface, or Calibration
- ◆ Versioning access in reconstruction
 - ◆ Default read access does 'the right thing'
 - ◆ Need mechanism to override the default
 - ◆ To recreate previous conditions
 - ◆ To select a particular version of a particular container explicitly
 - ◆ Reuse the concept of a 'configuration'?
- ◆ Conditions DB object tagging
 - ◆ Persistency (permanent, transitory, scratch, ...)
 - ◆ Need garbage collection
 - ◆ Use cases (Online, Offline, reference set, Physics analysis, ...)
 - ◆ DB subset creation tools
- ◆ Online interfaces are missing
 - ◆ Dataflow calibration (and configuration) server
 - ◆ PR distributed calibration and alignment object (not strictly conditions DB)