



CMS Calibration, Alignment & Databases Roadmap

Lee Lueking

May 17, 2005



General Info

- Calibration, Alignment, and Databases (CalAliDB) is now a real “box” under software. It is important to integrate closely with the other software areas.
- Frank Glege is responsible for online, and LL for offline. However, both On and Off will be viewed together, sharing solutions as much as possible.
- Oliver Buchmuller is my deputy at CERN to help with issues that need a physical presence there. Michael Case has helped in the past and he is always a great help.
- Fortnightly (bi-weekly) General meetings Tuesday @ 9 Chicago, 16:00 Geneva, time. Alternate Tuesdays will be used for special discussions (e.g. the Roadmap!).



The Road Map

- What is it for?
 - Collect technical discussions and reasoning in one place.
 - Help establish consensus on ultimate goals and design details.
 - Guide our development and allow us to work together, and in parallel.
 - Living document we will keep up to date as our understanding evolves.
 - Will be used as input for the Physics & Computing TDR's
- Other details
 - It is maintained in a CVS package, and can be updated by any of the “contributors”.
 - Frank, Oliver, and LL are the editors and will review the sections as they arrive.



Roadmap Chapters

1. Requirements and Use Cases. Overview of what is needed.
2. Sub-system needs descriptions. Separate documents, requirements and use cases includes online and offline needs for calibration/ alignment/ configuration/ trigger/ luminosity/... How and where they are produced and used. (Sub-system representatives)
3. Architecture and technology
4. DB environments and procedures. CVS, DB instances, development, testing, production. (Saima Iqbal, LL, CERN and FNAL DBAs)
5. Online detector specific schema designs descriptions and motivation. (Frank Glege)
6. Detector geometry database and alignment description and use. (Oliver Buchmuller)
7. Calibration overview and scale. This will be a summary of the sub-system reports. (Giacomo Bruno)



Roadmap Chapters (Cont.)

8. Generalized DB design and prototype for offline and some online sub-detectors. (LL Yuyi Guo, Gennadiy Lukhanin, Frank Glege)
9. Application DB API and associated machinery. Technical merits of various choices: RALCondDB AttributeList vs Strongly Typed objects, DB access technologies incl. Frontier, odbc, et cetera. Coordinate with relevant parts of EDM and ORCA to use provided Cal/Align/etc. (Based on report from API group)
10. Loading tools and access framework. Incl. Online to offline transfer mechanism. (LL , Frank Glege, Gennadiy L., Michael Case)
11. Delivery for distributed computing: HLT farm and Offline processing and analysis. (Lee, Emilio, Rep from Tier 1, Tier 2 center - Ian Fisk)
12. Deployment (LL, Frank)
13. GUI Framework evaluation and tools. (TBD)



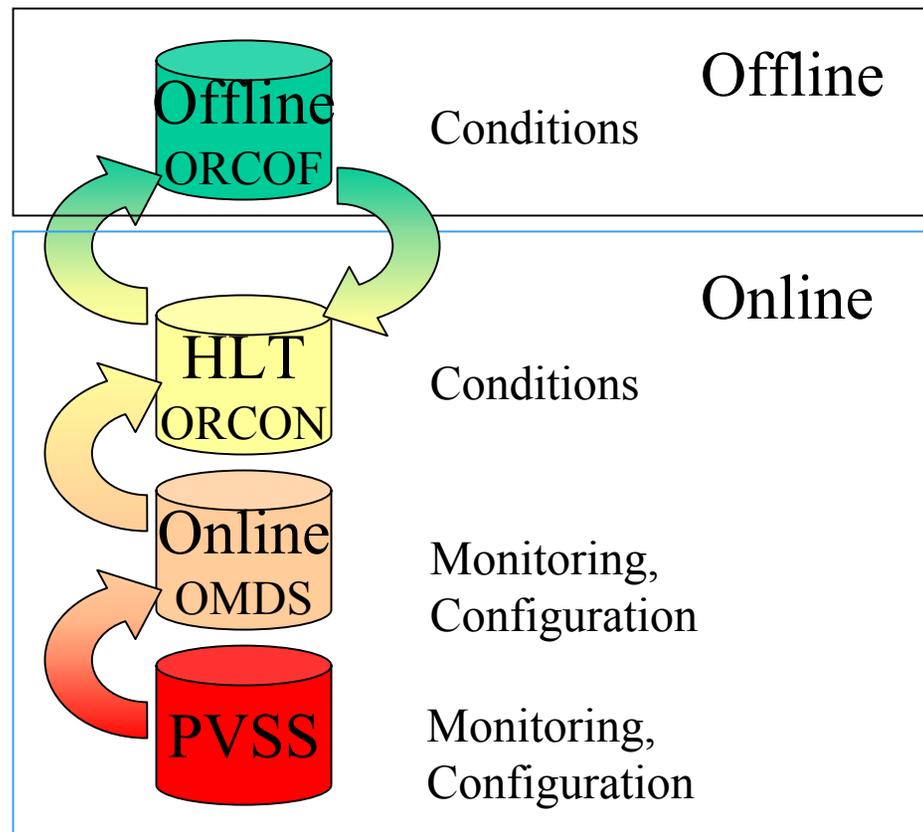
Outline for Sub-Detectors

1. A brief description of sub-system.
2. A description of how you plan to use the database (use cases or user narrative) for the cosmic challenge, detector commissioning, and physics running. (online and offline)
3. A summary of what data will be stored, number of channels, size (bytes), the anticipated frequency of change. Also, anticipated access patterns for each type of data. This includes geometry and alignment, hardware configuration, electronics run configuration, calibration, and monitoring. Summarizing this information in tables will be very useful. (online and offline)
4. A hierarchical overview of the database needs, and any schema designs you may already have.
5. Your anticipated schedule and milestones.
6. References to supporting documents.



Architecture and Technology

- Oracle is used for all production DB servers, both Online and Offline.
- Distributed access, HLT and Offline, to read-only data is through web-based proxy caching system (a la Frontier).
- Write, update, delete access for offline DB requires Grid-based authentication mechanism (a la LCG presumably).
- DB access is through C++ API, ROOT-ORACLE API, Java e.g. Object Relational Bridge (Apache OJB).



Conditions=calibration,alignment, slow controls data



Conditions DB: 3 Kinds of Conditions Data

- **Monitoring (DCS via PVSS)**
 - Sensor channel values (HV, LV, Temp, Pressure,...) move independently of each other in time.
 - Sensor locations have geometry scheme that is different from detector signal channels.
 - Data is collected in real time and has a single “*version*”.
 - IOV is the time region for which a value does not change.
- **Calibration**
 - Data loaded into database in “*sets*”, e.g. detector modules, super-modules, etc. There may be set overlaps.
 - An “*algorithm*” is used to create the data. More than one algorithm might be stored, and each might have multiple “*versions*”.
 - IOV is the time region for which a set is valid.
- **Alignment**
 - Similar to calibration. Sub detectors must be aligned relative each other.



Interval of Validity Management

- The monitoring, calibration, and alignment data is stored in the database, and accessed based on its Interval of Validity (IOV).
- The IOV is the range in time a value, or set of values, is valid for the running conditions when the data was taken.
- Since there are many IOVs spanning a long range of time, it is important to have a way of tagging them, similar to the tagging used for CVS code.
- In addition, the ability to have recursive tagging is needed so a chain, or tree, of related tags can be referred to easily.
- To maintain these tags, responsible persons from each sub-detector will be charged, and given authority to add and update the tags in the database related to their subsystem.



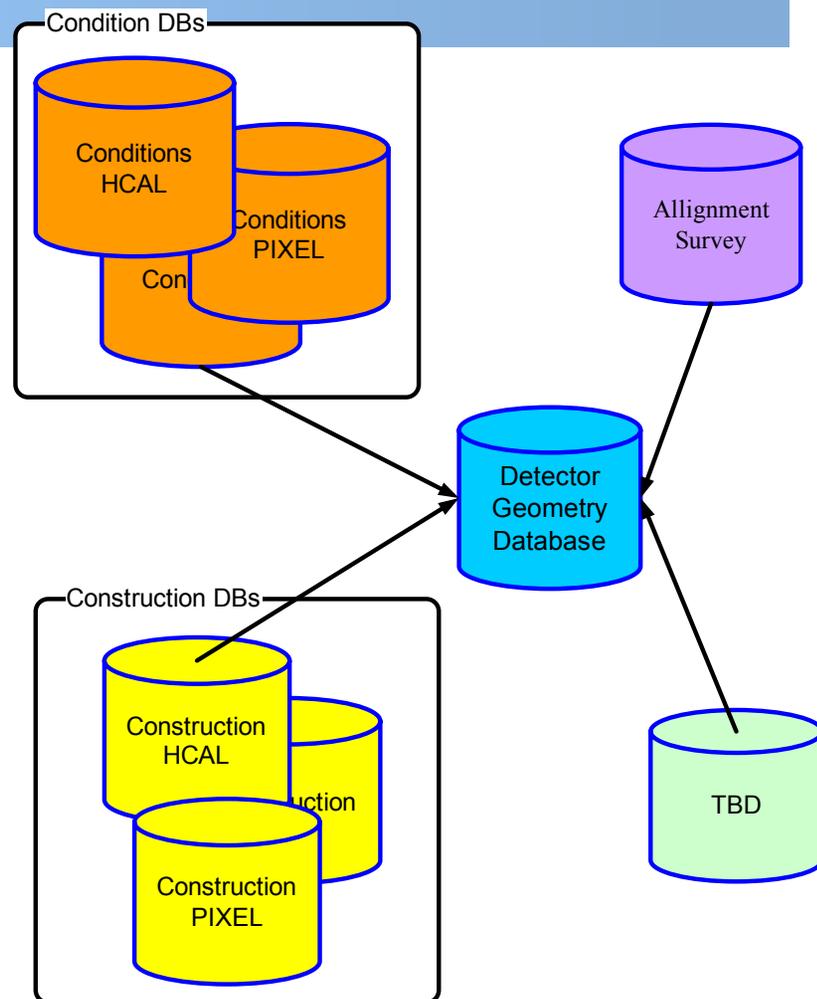
Database Environments & Procedures

- Operational Levels: Development, Integration, Production
- Naming conventions for Schema owners, and table-spaces.
- CVS hierarchy
- Bug reporting and change control: LCG Savanna
- Design tools: Oracle Designer recommended. Other tools (e.g. MS Visio is popular) used for prototyping.
- Schema review process
- Names and summary of dev, int, and prod database facilities at CERN and beyond.



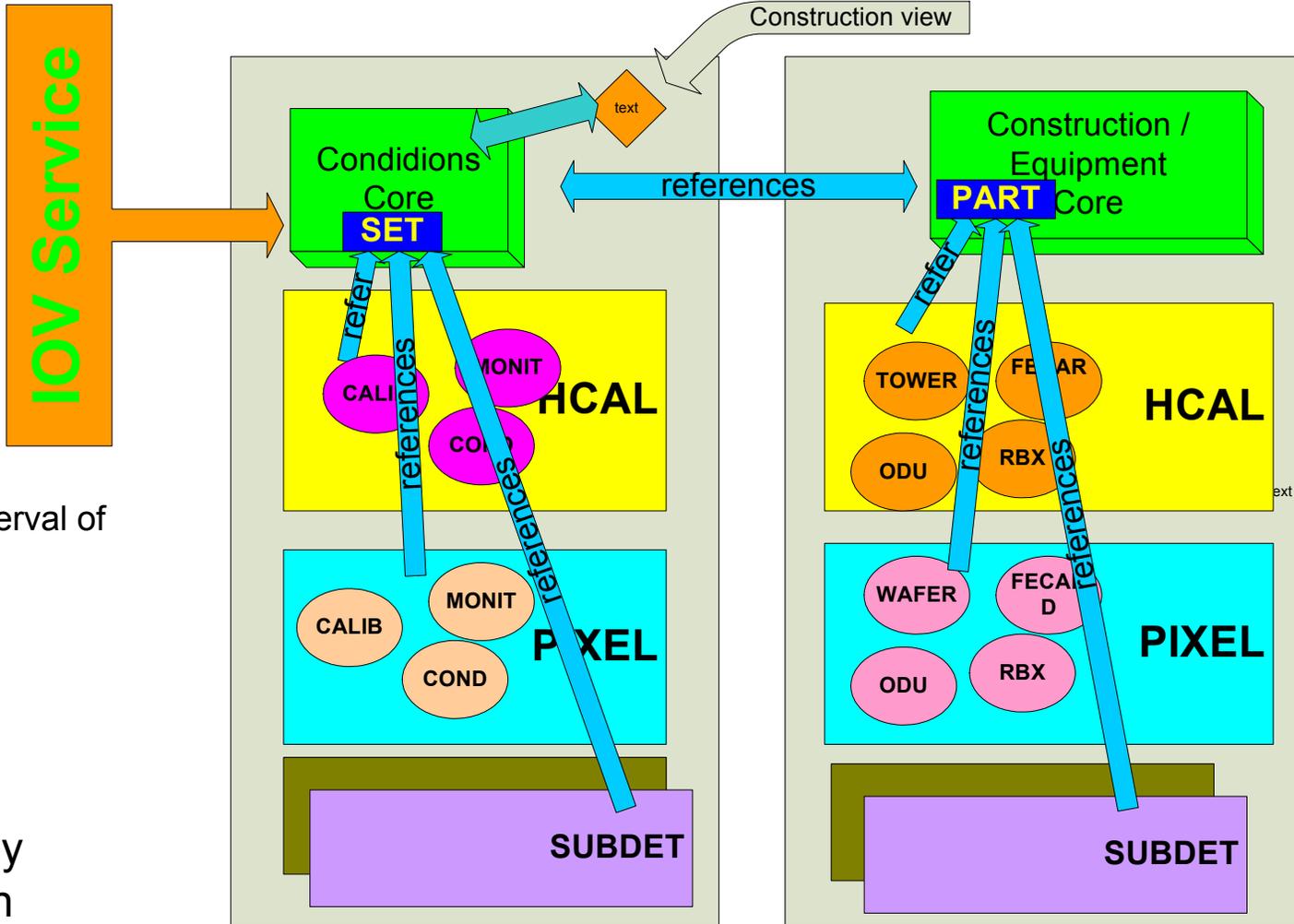
“Generalized” DB

- Generalized DB was proposed to as a common schema approach that could be used by all sub-detectors for offline.
- It uses the Detector Geometry Database as a starting point and extends it as suggested in CMS IN 2004/011.
- A prototype has been built and has some HCAL and PIXEL data in it for testing.





Generalized Database Schema



“IOV” = Interval of validity

Gennadiy Lukhanin



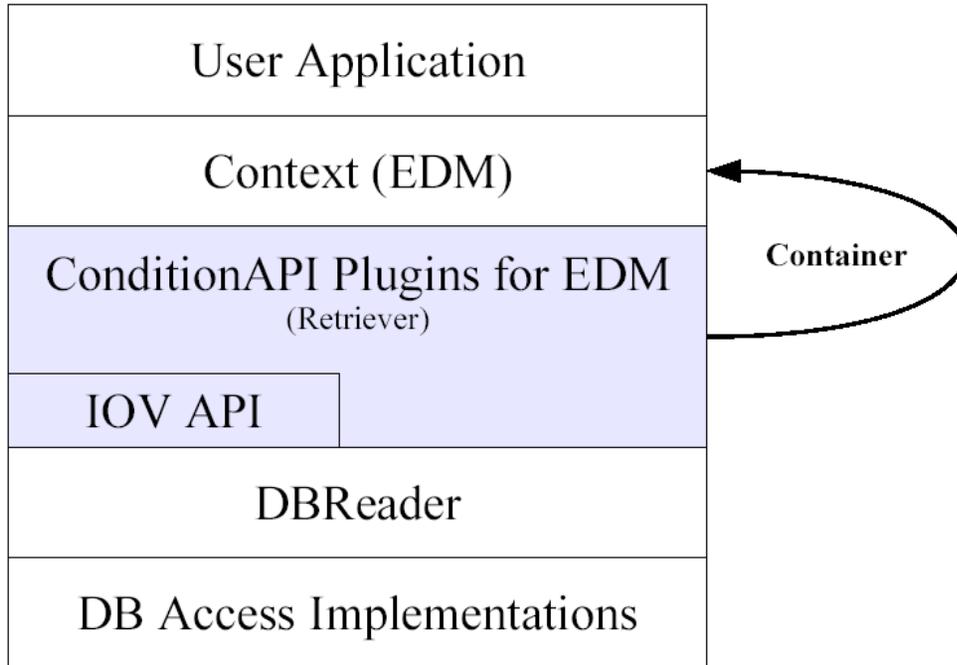
“Condition API” Extension

- Using strong typing (ST) for calibration and alignment objects is desirable. The existing ConditionAPI uses a dynamic approach referred to as Attribute List.
- A group was convened (Mid-April) to review options for ST and chart a strategy: Michael Case, Zhen Xie, Frank Glege, Sergey Kosyakov, Petar Maksimovic, Vincenzo Innocente, Chris Jones, Oliver Buchmuller. Others also participating incl. Lucia, Avi, Lee.
- Two options were proposed:
 - P1: An extension to ConditionAPI that assumes external code generation for the Object Relational (OR) mapping (a la CDF)
 - P2: A more POOL-centric approach with the OR mapping done within POOL -ORA (Object Relational Access) layer.
- Plug-ins for the EDM Context for each approach. Further comparison of features, liabilities, and performance will be done.

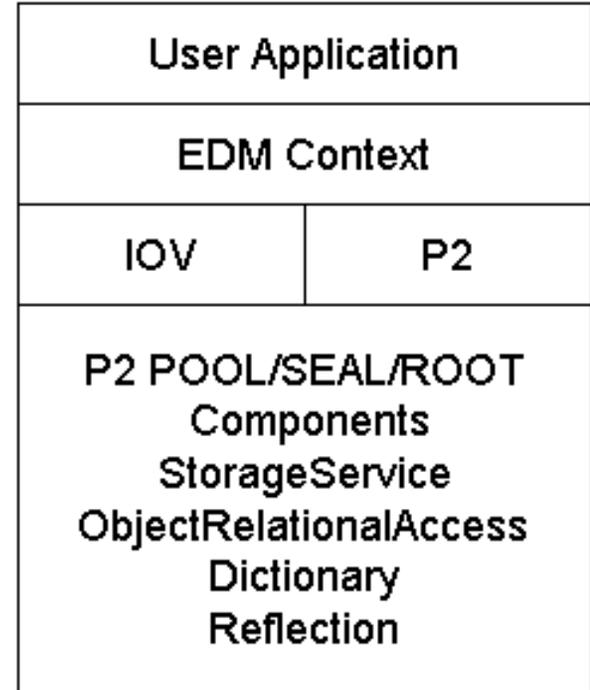


API proposals for Strongly Typed C++ Objects

Proposal 1



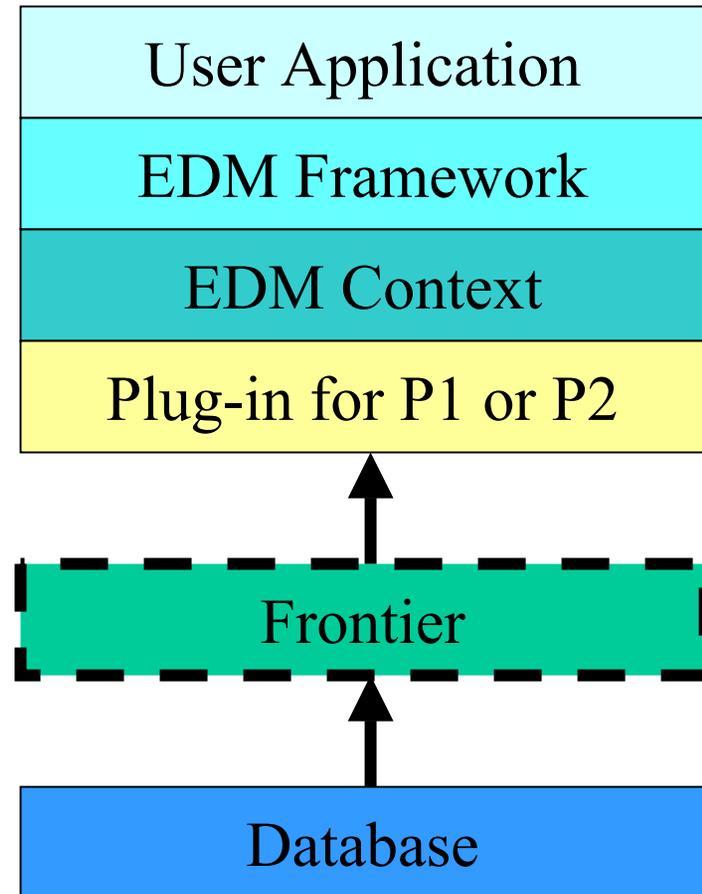
Proposal 2





End-to-end Test Chain

- End-to-end tests:
 - Access database for HCAL pedestals and gains
 - Access geometry from DGD (Detector Geometry DB) for tracker or other detectors.
 - Other tests possible, like accessing “simulated” PIXEL pedestals.





Database Access Technologies

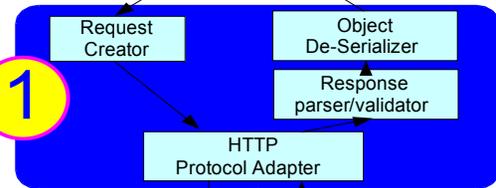
User Application

Frontier

2

Frontier Architecture

1

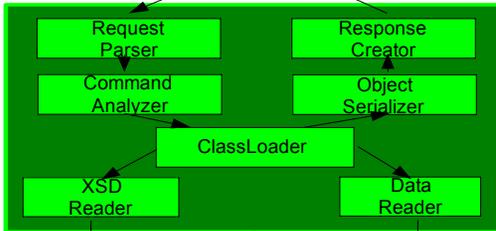


- FNAL product
- Used by CDF

IP Network

C++ Client API

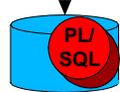
Squid Cache



ORCA Implementation

- R.Wilkinson (Caltech)
- S.Kosyakov (FNAL)

Tomcat Java AS



Frontier XSD Repository



JDBC-able Database

3

OO Java APIs, communicating with a database via O/R mapping technology



Oracle ROOT plug-in

Y. Liu (Iowa)

G. Lukhanin, Y.Guo (FNAL)



Access via Frontier

- This is the baseline offline distributed delivery system.
 - Frontier server will be deployed at CERN.
 - Squid proxy/caches deployed at each Tier 1 and Tier 2 center. ACL (Access Control Lists) configured to accommodate needed users/servers.
- Operating with large objects (100s of MBs) has been tested and works well.
- Could be used for HLT farm online.



HLT Frontier Testing

- Testing will be performed on the existing HLT filter farm to verify throughput and reliability.
 - Squid server load test
 - Full Frontier test (server + squid + client).
- Initial specs for the test are in the Roadmap



Root-Oracle Plugin

How to use:

- Download and compile the latest version of ROOT package;
- Register your database connection in TNSNAMES.ORA file;
- Set the following environment variables:

```
export ROOTSYS=/opt/root.db
export LD_LIBRARY_PATH=
$ROOTSYS/lib/root:/afs/cern.ch/project/oracle/instantclient/10.1.0.2/rh73_gcc323/lib:${LD_LIBRARY_PATH}
export PATH=$ROOTSYS/bin:${PATH}
export TNS_ADMIN=$ROOTSYS
```

- Create a SQL query;
- Update a sample script to match your query;

```
#include "conn_db.h"
#include <iostream.h>
void ResponcePlot(char* barcode ="304000000000100748",
int position = 2){
char * sqlStatement =
"select \
    ADC_CAHANNEL_NUMBER, CAPACITOR_0_VALUE \
from \
    COND_DATA_SETS \
where \
    BARCODE =\ '%s\ ' and POSITION=\ '%d\ '";
char sql[10000];
sprintf(sql,sqlStatement,barcode,position);

//connect with oracle server
TSQLServer *db=TSQServer::Connect(DB_PATH,DB_USER,DB_PASS);
//do the query
TSQLResult *res=db->Query(sql);
TSQLRow *rowTSQL=res->Next();
TOracleRow * row=(TOracleRow *)rowTSQL;
/*get rows one by one */
UInt_t chan[128];Float_t capid1[128];int j=0;
do{
    chan[j]=*(UInt_t*)row->GetField(0);
    capid1[j]=*(Float_t*)row->GetField(1);
    j++;
}while (res->Next() && j<128);
TCanvas *c1=new TCanvas("c1","Plot",5);c1->Divide(2,2);
TGraph *gr1=new TGraph(j,chan,capid1);
gr1->SetTitle("PLOT");c1->cd(1);gr1->Draw("ALP");
c1->Print("ROOT_Plot.eps","eps");
delete row;delete res;delete db;
}
```



Schedule

- May 2 – First draft of Roadmap. Finalize plan for C++ API and DB access machinery (continued testing in end-to-end).
- May 15 – Integration and production environments, exercise shortly after that. Understand online to offline transfer needs. End to end test (DB to EDM framework) with HCAL (existing TB calib schema), and tracker geometry. Begin Frontier testing with HLT filter farm. GenDB approach understood and agreed to by sub-detector groups.
- June 1 – Generalized schema prototyping completed for PIXEL and HCAL. Work with additional detectors to develop and exercise automation machinery and interfaces.
- June 15 – First example of online to offline transfer tool. New GenDB schema loaded with HCAL TB calibration.
- July-August – Graphical tools. (PIXEL construction tools may be ready prior to this.) Deploy Gen Schema to production at CERN. Begin populating with subsystem information as available. Exercise DB access to reconstruction and HLT framework.
- September 15 – Most sub-systems ready for testing in Gen DB. Begin load testing. Start integration for Cosmic Challenge.



The End