lcsim Conditions System

Tony Johnson
tonyj@slac.stanford.edu

Design

- Separate user interface from implementation
- Separate user interface and implementation from conditions store
  - Conditions objects presented to user are independent of conditions store
- Allow arbitrary, extensible set of conditions
- Thread safe for multi-core implementation
- Conditions dependent on detector/geometry/time (run)
  - Ready for ILC running in 2005!

Current Implementation

- Current *implementation* is biased towards lcsim’s earlier use for detector simulation
  - Run/time dependence mostly ignored
  - Deliberate bias against using databases
    - Support for running with no internet, local databases
  - All conditions packaged in detector/geometry dependent zip files
    - Downloaded on-demand and cached locally
    - Zip files typically contain
      - Property files – simple key value pairs with native Java support
      - XML files – simple to parse using Java’s built-in XML support
      - Other format files – require user-written access classes

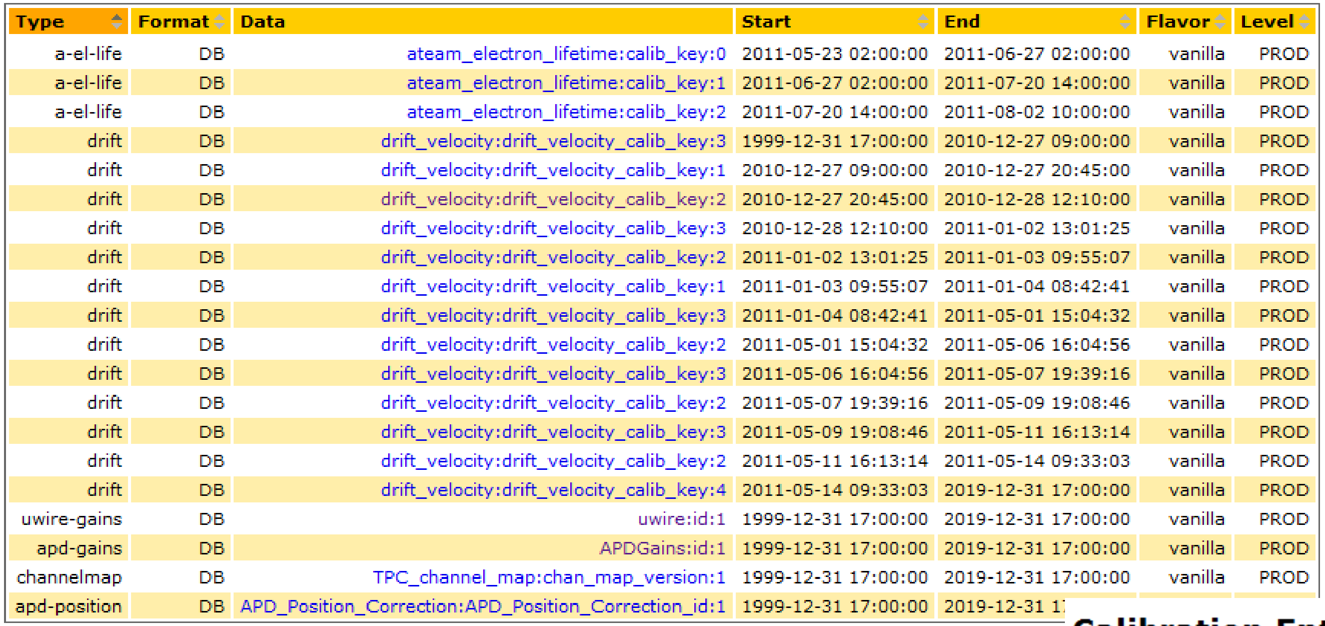
User Interface

- Simple Use:
  - ConditionsManager manager = ConditionsManager.defaultInstance();
  - manager.setRun(run); // called by framework
  - ConditionsSet conditions = manager.getConditions(“DriftConstants”);
  - Distance d = conditions.getDouble(“drift velocity”)*driftTime;
- ConditionsSet will be updated automatically as run number changes
  - If you want to be notified of changes
    - Conditions.addConditionsListener(myListener);
- ConditionsSet represents set of key/value pairs
  - Suitable for reading from properties file or SQL table
  - Most common type of conditions object
- Also support:
  - CachedConditions<T> – wrapper around arbitrary object
    - Requires a converter to be registered with conditions manager
    - manager.registerConditionsConverter(myConditionsConverter)
  - SubConditions – allows conditions to contain nested sub-conditions

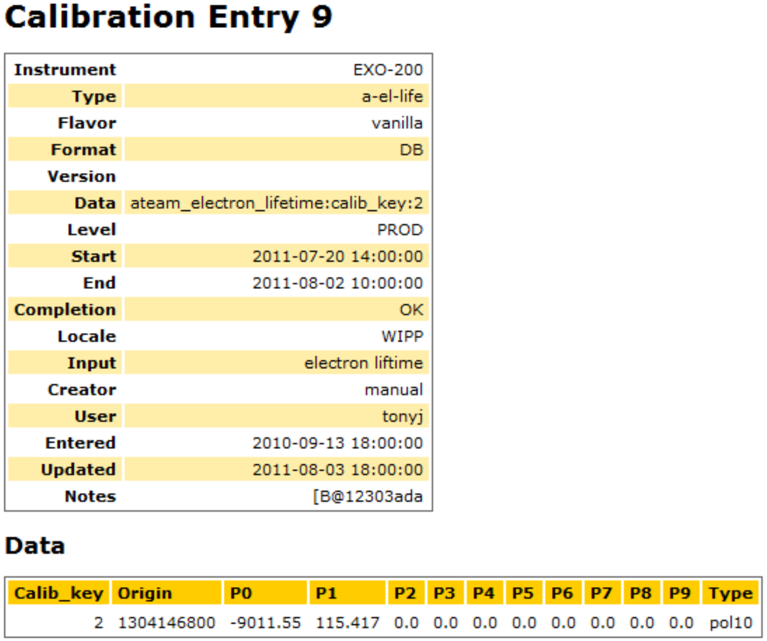
Detour – EXO conditions database

- Based on Fermi calibration database
  - Minimalized for use by small, manpower limited experiment
  - Implemented by Joanne Bogart
    - Who also implemented Fermi Conditions System
    - C++ implementation took Joanne a few weeks
      - Significant fraction of this was learning EXO framework, implementing low level database functionality which comes for free with Java
    - Implemented as mysql database hosted by SLAC computing division
      - (read-only) access from anywhere
  - Flexible design, allows conditions data itself to be stored in database itself, or in external files in any format
    - So far all EXO conditions live in database
  - Simple web interface

EXO Conditions database



Type	Format	Data	Start	End	Flavor	Level
a-el-life	DB	ateam_electron_lifetime:calib_key:0	2011-05-23 02:00:00	2011-06-27 02:00:00	vanilla	PROD
a-el-life	DB	ateam_electron_lifetime:calib_key:1	2011-06-27 02:00:00	2011-07-20 14:00:00	vanilla	PROD
a-el-life	DB	ateam_electron_lifetime:calib_key:2	2011-07-20 14:00:00	2011-08-02 10:00:00	vanilla	PROD
drift	DB	drift_velocity:drift_velocity_calib_key:3	1999-12-31 17:00:00	2010-12-27 09:00:00	vanilla	PROD
drift	DB	drift_velocity:drift_velocity_calib_key:1	2010-12-27 09:00:00	2010-12-27 20:45:00	vanilla	PROD
drift	DB	drift_velocity:drift_velocity_calib_key:2	2010-12-27 20:45:00	2010-12-28 12:10:00	vanilla	PROD
drift	DB	drift_velocity:drift_velocity_calib_key:3	2010-12-28 12:10:00	2011-01-02 13:01:25	vanilla	PROD
drift	DB	drift_velocity:drift_velocity_calib_key:2	2011-01-02 13:01:25	2011-01-03 09:55:07	vanilla	PROD
drift	DB	drift_velocity:drift_velocity_calib_key:1	2011-01-03 09:55:07	2011-01-04 08:42:41	vanilla	PROD
drift	DB	drift_velocity:drift_velocity_calib_key:3	2011-01-04 08:42:41	2011-05-01 15:04:32	vanilla	PROD
drift	DB	drift_velocity:drift_velocity_calib_key:2	2011-05-01 15:04:32	2011-05-06 16:04:56	vanilla	PROD
drift	DB	drift_velocity:drift_velocity_calib_key:3	2011-05-06 16:04:56	2011-05-07 19:39:16	vanilla	PROD
drift	DB	drift_velocity:drift_velocity_calib_key:2	2011-05-07 19:39:16	2011-05-09 19:08:46	vanilla	PROD
drift	DB	drift_velocity:drift_velocity_calib_key:3	2011-05-09 19:08:46	2011-05-11 16:13:14	vanilla	PROD
drift	DB	drift_velocity:drift_velocity_calib_key:2	2011-05-11 16:13:14	2011-05-14 09:33:03	vanilla	PROD
drift	DB	drift_velocity:drift_velocity_calib_key:4	2011-05-14 09:33:03	2019-12-31 17:00:00	vanilla	PROD
uwire-gains	DB	uwire:id:1	1999-12-31 17:00:00	2019-12-31 17:00:00	vanilla	PROD
apd-gains	DB	APDGains:id:1	1999-12-31 17:00:00	2019-12-31 17:00:00	vanilla	PROD
channelmap	DB	TPC_channel_map:chan_map_version:1	1999-12-31 17:00:00	2019-12-31 17:00:00	vanilla	PROD
apd-position	DB	APD_Position_Correction:APD_Position_Correction_id:1	1999-12-31 17:00:00	2019-12-31 1		



Calibration Entry 9

Instrument	EXO-200
Type	a-el-life
Flavor	vanilla
Format	DB
Version	
Data	ateam_electron_lifetime:calib_key:2
Level	PROD
Start	2011-07-20 14:00:00
End	2011-08-02 10:00:00
Completion	OK
Locale	WIPP
Input	electron liftime
Creator	manual
User	tonyj
Entered	2010-09-13 18:00:00
Updated	2011-08-03 18:00:00
Notes	[B@12303ada

Data

Calib_key	Origin	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	Type
2	1304146800	-9011.55	115.417	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	pol10

Possible way forward

- Get started using existing implementation
  - Conditions stored in properties, xml, etc files in .zip file
- Set up conditions database modeled on Fermi/EXO system
  - Can be hosted at SLAC computer center (mysql)
  - And/or Jlab perhaps using mysql mirroring
- Create new conditions implementation
  - Use Java JDBC to interface to database
  - Possible small tweaks to user interface
  - A few weeks work (for someone)
    - lcsim Conditions/EXO database look to be very compatible
    - Run vs Time issue

Another Detour – Thread Safety

- Java has build in support for thread safety
  - Especially in multi-core/online environment one may want to be able to process several events in parallel
- lcsim was originally designed to be compatible with multi thread operation but
  - Time/Manpower pressure
  - Inexperience developers
  - Desire to simplify things for users/developers
- Resulted in many “singletons” which make thread safety more difficult (although not impossible)
    - ConditionsManager.defaultInstance() // Dangerous!
    - event.getConditionsManager() // Good, but not currently available
  - This may need some review/refactoring for online use

Links

- https://confluence.slac.stanford.edu/display/ilc/Conditions+Database+Overview
  - Heavily biased to current implementation
- http://www.lcsim.org/software/lcsim/1.13/apidocs/org/lcsim/conditions/package-summary.html
  - Not most recent version?
- https://confluence.slac.stanford.edu/display/exo/Calibration+Software
  - Restricted to EXO members – perhaps we can make a copy