New Developments for iLCSoft: iLCTest and LCCD

Steve Aplin and Jan Engels

International Workshop on Linear Colliders 2010
19th October 2010

Overview

- Recent Improvements to LCCD
  - Default Collections
  - Folder Tagging
  - LCCD Exceptions
- iLCTest
  - A CTest and CDash based testing system for iLCSoft

Overview

Linear **C**ollider **C**onditions **D**ata Toolkit

- Supports test-beam efforts by meeting the need to store and retrieve conditions data, e.g. slow control, electronics setup and calibration constants.
- LCCD provides a toolkit that allows conditions data to be stored either in a Database or within an LCIO file in a transparent way.
- The main purpose is to provide an easy to use interface to read conditions data in any program that analyzes LCIO data.

- Current Release – v01-01
- Currently used by Calice and LC-TPC

Overview

Linear **C**ollider **C**onditions **D**ata Toolkit

4 different use cases implemented in LCCD:

- Read conditions data that is valid for the time specified by the time stamp in the current LCEvent on the fly from a data base.
- Read one particular set of conditions data, e.g. calibration constants from an LCIO file in a small job.
- Read conditions data that occurs in the data stream of an LCIO file.
- Read conditions data from an LCIO file that has been created such that it holds the data for a given time range with consecutive validity intervals in consecutive events.

Overview

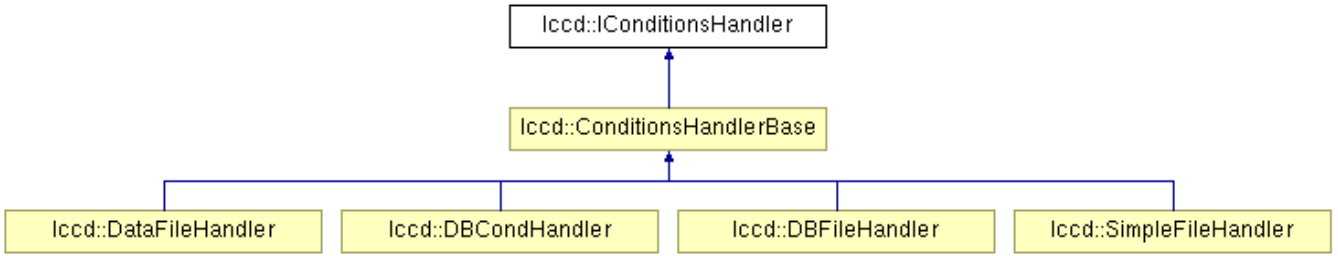
Linear **C**ollider **C**onditions **D**ata Toolkit

- lccd::DBInterface provides easy to use methods to store and tag the data in a conditions database.
- The only requirement is that the data is available in an LCCollection of LCGenericObject subclasses.
- The data itself is stored as a BLOB (binary large object) in the database.

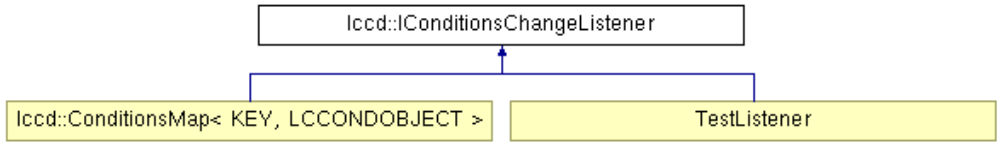
Overview

Linear **C**ollider **C**onditions **D**ata Toolkit

Inheritance diagram for lccd::IConditionsHandler:

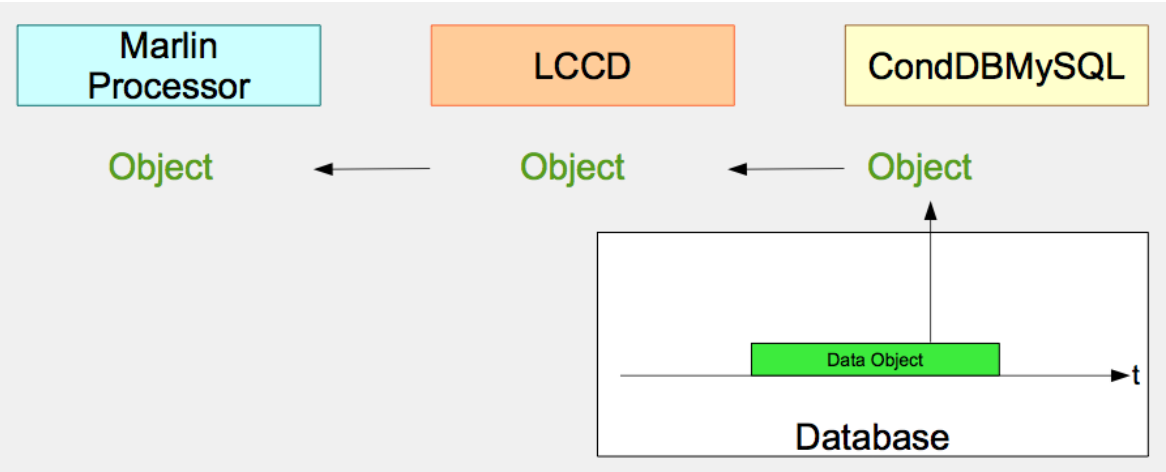


Inheritance diagram for lccd::IConditionsChangeListener:



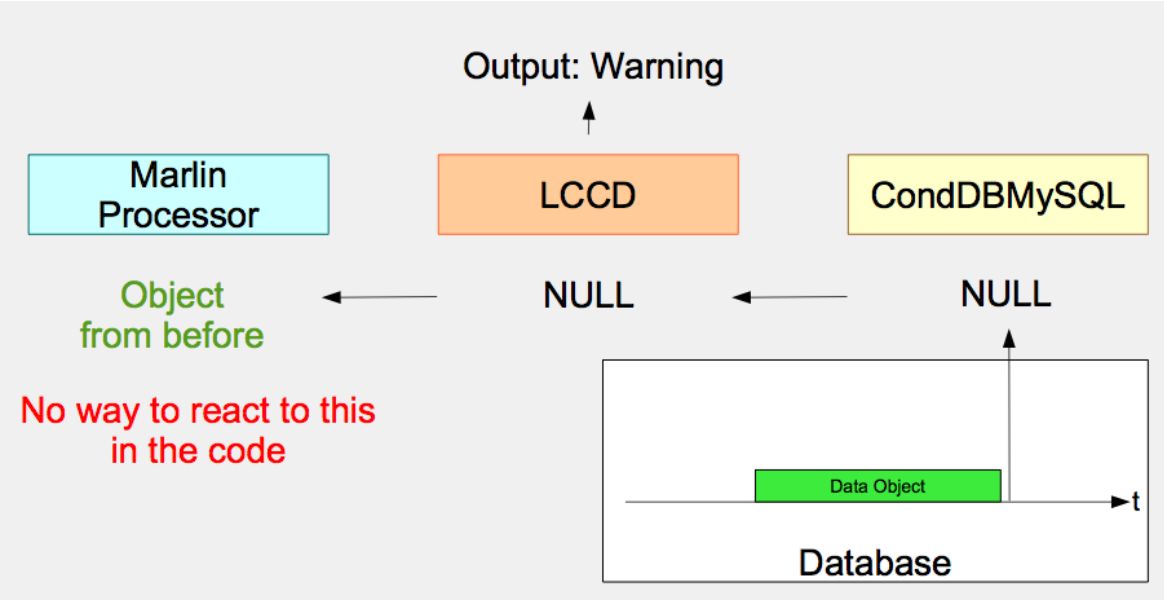
Default Collections

- Originally LCCD did not foresee valid regions of time where no collection stored.



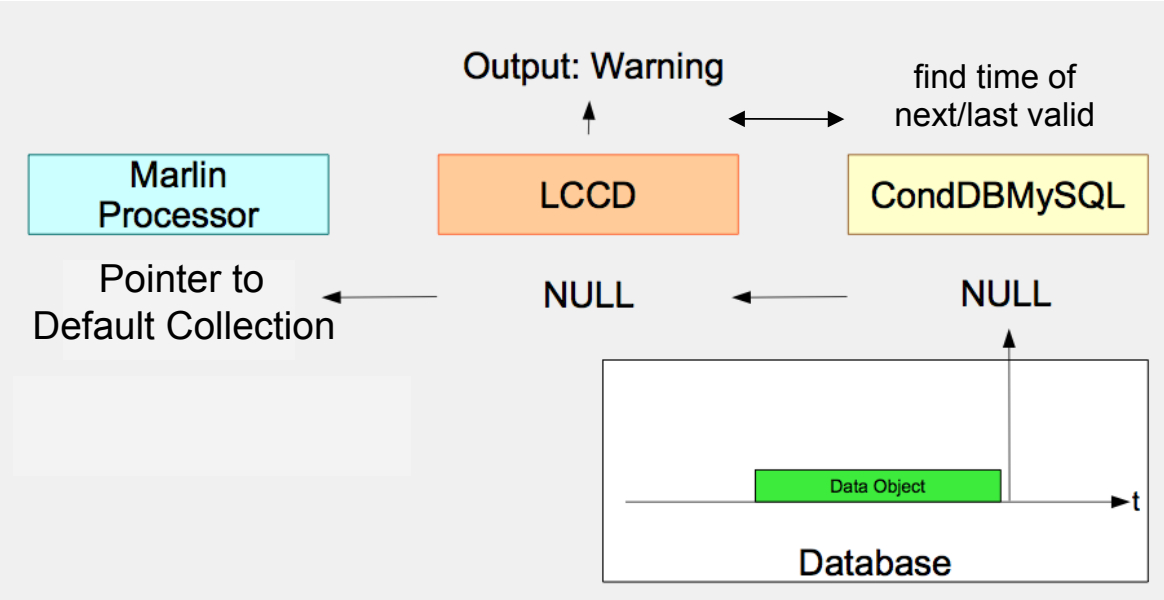
Default Collections

- LCCD was then modified to catch the exception for the case of no collections found so as to allow further processing.
- Due to the use of the Listener mechanism, this meant that the Marlin Processors were now blinded to real problems with missing collections.
- As a consequence of missing collections, this lead to very high DB load.



Default Collections

- LCCD interface has now been extended to allow users to register a Default Collection which will be returned if no valid collection is found in the Data Base or DBFile.
- IConditionsChangeListener now contains the two additional call back methods:



Default Collections

- LCCD interface has now been extended to allow users to register a Default Collection which will be returned if no valid collection is found in the Data Base or DBFile.
- IConditionsChangeListener now contains the two additional call back methods:
  - virtual void registeredWithHandler(IConditionsHandler* ch) ;
  - virtual void deRegisteredWithHandler(IConditionsHandler* ch) ;
- These are used to maintain a std::list of pointers to the handlers with which the listener has been registered.
- Note: this functionality is only implemented in the DBCondHandler and DBFileHandler classes.
  - Using these methods with SimpleFileHandler and DataFileHandler classes will cause an exception to be thrown.

Default Collections

- The ConditionsHandlerBase class has been declared a friend class of IConditionsChangeListener and uses the call-back methods when a listener is registered or de-registered respectively.
- IConditionsHandler has also been extended to provide pointers to the default collection and the last valid collection.
  - virtual lcio::LCCollection* defaultCollection() = 0 ;
  - virtual lcio::LCCollection* lastValidCollection() = 0;
- IConditionsHandler has also been extended to check if a given IConditionsChangeListener is register with it
  - virtual bool isChangeListenerRegistered(IConditionsChangeListener* cl) ;

Default Collections

```
SimpleListener::SimpleListener(){
  std::cout << "SimpleListener::SimpleListener()" << std::endl;

  // create an empty collection for this listener: later this could be a global for all listeners
   _myEmptyCollection = new LCCollectionVec( LCIO::LCGENERICOBJECT );
   _myEmptyCollection->parameters().setValue("CollectionName", "this is myEmptyCollection" ) ;
}


void SimpleListener::conditionsChanged( lcio::LCCollection* col ){

  std::cout << "SimpleListener::conditionsChanged()" << std::endl;

  // look into the map to see if we have accepted this collection as a default
  std::map<lcio::LCCollection* ,lccd::IConditionsHandler* >::iterator it = _handlerDefaultCollecionMap.find(col);

  // check if the collection is our default collection
  if ( it != _handlerDefaultCollecionMap.end()) {
    std::cout << "SimpleListener::conditionsChanged(): default collection sent" << std::endl;
    std::cout << "SimpleListener::conditionsChanged(): CollectionName: " << col->getParameters().getStringVal( "CollectionName" ) << std::endl;
  }
  else { // it is not a default so we can do anything we like
    std::cout << "SimpleListener::conditionsChanged(): CollectionName: " << col->getParameters().getStringVal( "CollectionName" ) << std::endl;
  }

}
```

Default Collections

```
void SimpleListener::registeredWithHandler( lccd::IConditionsHandler* ch ){

  std::cout << "SimpleListener::registeredWithHandler(): registered with:" << ch->name() << std::endl;

  std::cout << "SimpleListener::registeredWithHandler(): register default collection:" << std::endl;
  // try to get the default collection
  LCCollection* col = ch->defaultCollection();


  if( ! col ){ // it will be null if none has so far been registered. So let's register ours
    ch->registerDefaultCollection( _myEmptyCollection );
    std::cout << "SimpleListener::registeredWithHandler(): default collection registered:" << std::endl;
    // and put in the map for this handler
    _handlerDefaultCollecionMap[_myEmptyCollection] = ch;
  }

  else if( col == _myEmptyCollection ){ // then the default handler was already registered, that's odd ... ;)
    std::cout << "SimpleListener::registeredWithHandler(): default collection is already set to myEmptyCollection:" << std::endl;
  }

  else { // somebody has got there before us, let's see if we like the default ...

    // here well look at the collections name to see if we like it
    lcio::StringVec StringKeys;
    StringKeys = col->getParameters().getStringKeys(StringKeys);
    for( unsigned int i=0; i<StringKeys.size();++i ){
        if( StringKeys.at(i) == "CollectionName" && col->getParameters().getStringVal(StringKeys.at(i)) == "I am empty" ) {
            std::cout << "SimpleListener::registeredWithHandler(): I like your default ;)" << std::endl;
            _handlerDefaultCollecionMap[col] = ch;
          }
          else{
            std::cout << "SimpleListener::registeredWithHandler(): I don't like your default, leave my handler alone ;)" << std::endl;
            throw std::exception();
          }
        }
    }
}
```

Folder Tagging

- Previously not possible to tag a folder with a **tag** which has been used to tag another folder.
- To solve this, a recursive search is now done when trying to tag a folder. This checks if the desired **tag** has been already used for the folder in question, or for any of its sub-folders.
- If the **tag** is found in the folder branch by the recursive search an exception is thrown and no tagging is performed.

Ralf Diener

LCCD Exceptions

- Similar to those defined in LCIO
- Part of the lccd namespace

```
class LCCDException : public std::exception

 LCCDException( const std::string& text ) {
  message = "lccd::Exception: " + text ;
 }

 DatabaseException( std::string text ){
  message = "lccd::DatabaseException: " + text ;
 }

 DataNotAvailableException( std::string text ) {
  message = "lccd::DataNotAvailableException: " + text ;
 }

 ReadOnlyException( std::string text ){
  message = "lccd::ReadOnlyException: " + text ;
 }

 InconsistencyException( std::string text ) {
  message = "lccd::InconsistencyException: " + text ;
 }

 MemberNotImplementedException( std::string text ) {
  message = "lccd::MemberNotImplementedException: " + text ;
 }
```

Welcome improvement
in terms of error handling

Ralf Diener

Summary of Improvements to LCCD

- Default Collections – available since v01-00
- Folder Tagging – to be available from v01-01 *
- LCCD Exceptions – to be available from v01-01
- Next Release v01-01 – within iLCSoft v01-10

* needs release of CondDBMySQL_ILC-0-9-1

iLCTest

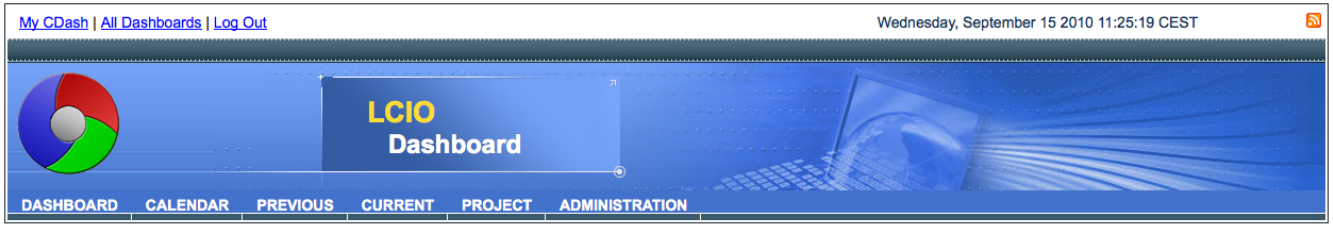
- Ctest & CDash based test system running since June
- Software Package Integration tests established
- Infrastructure code developed to make adding new tests straight-forward

CDash Overview

My Projects

Project Name	Actions	Builds	Builds per day	Success Last 24h	Errors Last 24h	Warnings Last 24h
Calice		115	5	4	0	1
CED		614	6	6	0	0
CEDViewer		614	6	0	0	6
GEAR		618	6	6	0	0
ilcinstall		202	2	0	0	2
ILCTest		11	0	0	0	0
LCCD		615	6	3	0	3
LCIO		618	6	0	0	6
Marlin		625	6	0	0	6
MarlinReco		618	6	0	0	6
MarlinUtil		614	6	6	0	0
Overlay		614	6	6	0	0
RAIDA		614	6	6	0	0

e.g. LCIO



No file changed as of **Wednesday, September 15 2010 00:00:00 CEST**

Help

[Show Filters]

Nightly

Site	Build Name	Update		Configure			Build			Test				Build Time
		Files	Min	Error	Warn	Min	Error	Warn	Min	NotRun	Fail	Pass	Min	
grid-ilc-pa0	linux-gcc-debug			0	0	0	0	12	0.1					2010-09-15T02:01:57 CEST
grid-ilc-pa0	linux-gcc-debug-x64			0	0	0	0	12	0.1					2010-09-15T04:01:42 CEST
grid-ilc-pa0	linux-gcc-default			0	0	0	0	12	0.2					2010-09-15T02:02:08 CEST
grid-ilc-pa0	linux-gcc-default-tests	0	0.1	0	0	0	0	13	0.3	0	0	21	0.3	2010-09-15T02:01:11 CEST
grid-ilc-pa0	linux-gcc-default-tests-x64	0	0.1	0	0	0	0	13	0.2	0	0	21	0.2	2010-09-15T04:01:08 CEST
grid-ilc-pa0	linux-gcc-default-x64			0	0	0	0	12	0.2					2010-09-15T04:01:54 CEST
Totals	**6 Builds**	0	0.2	**0**	**0**	0	**0**	**74**	1.1	**0**	**0**	**42**	0.5	

No Continuous Builds

Name	Status	Time Status	Time	Details
t_c_ana_c2j_rec	Passed	Passed	0.14	Completed
t_c_ana_c_rec	Passed	Passed	0.17	Completed
t_c_ana_c_sim	Passed	Passed	0.09	Completed
t_c_ana_j2c_rec	Passed	Passed	0.13	Completed
t_c_ana_j_rec	Passed	Passed	0.10	Completed
t_c_ana_j_sim	Passed	Passed	0.06	Completed
t_c_rec_c_sim	Passed	Passed	3.78	Completed
t_c_rec_j_sim	Passed	Passed	1.95	Completed
t_c_sim	Passed	Passed	1.99	Completed
t_j_ana_c_rec	Passed	Passed	1.88	Completed
t_j_ana_c_sim	Passed	Passed	1.46	Completed
t_j_ana_j_rec	Passed	Passed	1.41	Completed
t_j_ana_j_sim	Passed	Passed	0.87	Completed
t_j_rec_c_sim	Passed	Passed	2.03	Completed
t_j_rec_j_sim	Passed	Passed	1.63	Completed
t_j_sim	Passed	Passed	0.65	Completed
t_test_calohit	Passed	Passed	0.14	Completed
t_test_example	Passed	Passed	0.00	Completed
t_test_randomaccess	Passed	Passed	0.01	Completed
t_test_trackerhit	Passed	Passed	0.10	Completed
t_test_trackerpulse	Passed	Passed	0.10	Completed

CMake tests defined in LCIO are automatically published to CDash on a nightly basis by CTest

iLCSoft Integration Tests Overview

History

[Show Build History]
[Build History Filter]

Date	Update Files	Update Errors	Update Warnings	Configure Errors	Configure Warnings	Build Errors	Build Warnings	Tests Failed
2010-09-15 03:01:15	0	0	0	0	0	0	50	0
2010-09-14 03:01:14	0	0	0	0	0	0	50	0
2010-09-13 03:01:13	0	0	0	0	0	9	50	0
2010-09-12 03:01:12	0	0	0	0	0	9	50	0
2010-09-11 03:01:11	0	0	0	0	0	5	50	0
2010-09-10 03:01:10	0	0	0	0	0	9	50	0
2010-09-09 03:01:09	0	0	0	0	0	9	50	0
2010-09-08 03:01:08	0	0	0	0	0	0	50	0
2010-09-07 03:01:07	0	0	0	0	0	8	50	5
2010-09-06 03:01:06	0	0	0	0	0	0	50	0
2010-09-05 03:01:05	0	0	0	0	0	0	50	0
2010-09-04 03:01:04	0	0	0	0	0	0	50	0
2010-09-03 03:01:03	0	0	0	0	0	0	50	0
2010-09-02 03:01:02	0	0	0	0	0	0	50	0
2010-09-01 03:01:01	0	0	0	0	0	0	50	0
2010-08-31 03:01:31	0	0	0	0	0	0	50	0
2010-08-30 03:01:30	0	0	0	0	0	0	50	0
2010-08-29 03:01:29	0	0	0	0	0	0	50	0
2010-08-28 03:01:28	0	0	0	0	0	0	50	0
2010-08-27 03:01:27	0	0	0	0	0	0	50	0
2010-08-26 03:01:26	0	0	0	0	0	0	50	0
2010-08-25 03:01:25	0	0	0	0	0	0	50	0
2010-08-24 03:01:24	0	0	0	0	0	0	50	0
2010-08-23 03:01:23	0	0	0	0	0	22	50	5
2010-08-22 03:01:22	0	0	0	0	0	22	50	5
2010-08-21 03:01:21	0	0	0	0	0	22	50	5
2010-08-20 03:01:20	0	0	0	0	0	0	50	0
2010-08-19 03:01:19	0	0	0	0	0	0	50	0

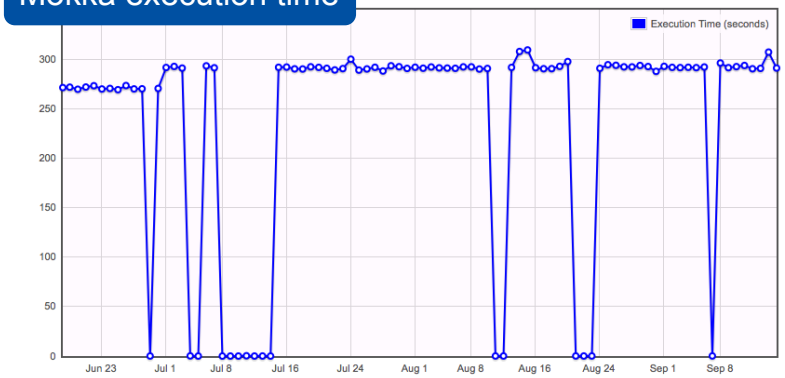
On a Nightly basis runs the Standard LOI reconstruction over a handful of events to test that the full Sim/Rec software chain is working properly.

Automatic Email notification if and when build or tests fail.

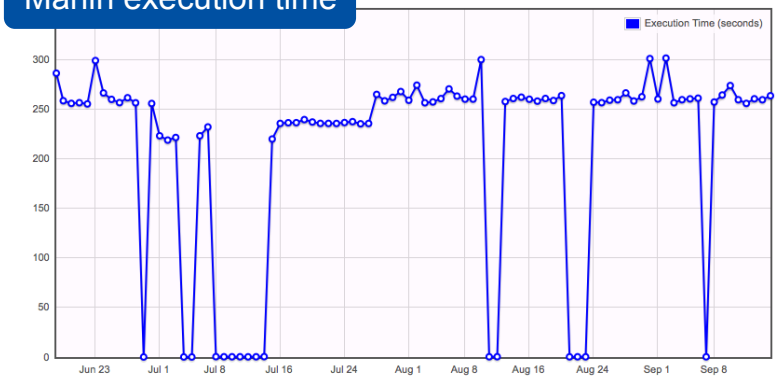
Developers may subscribe to several projects to keep an overview

iLCSoft Integration Tests Detailed

Mokka execution time

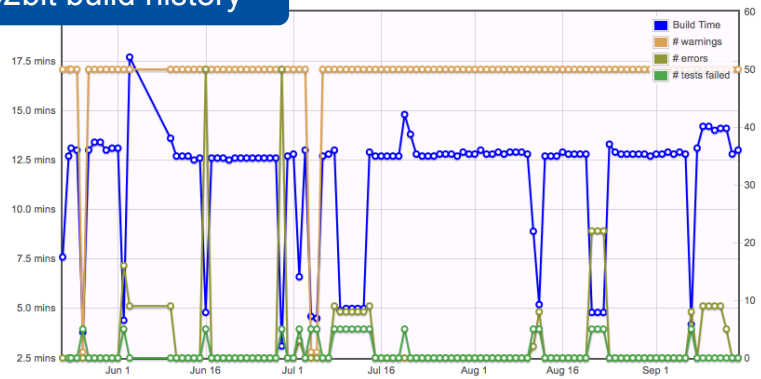


Marlin execution time



On a Nightly basis runs the Standard LOI reconstruction over a handful of events to test that the full Sim/Rec software chain is working properly.

32bit build history

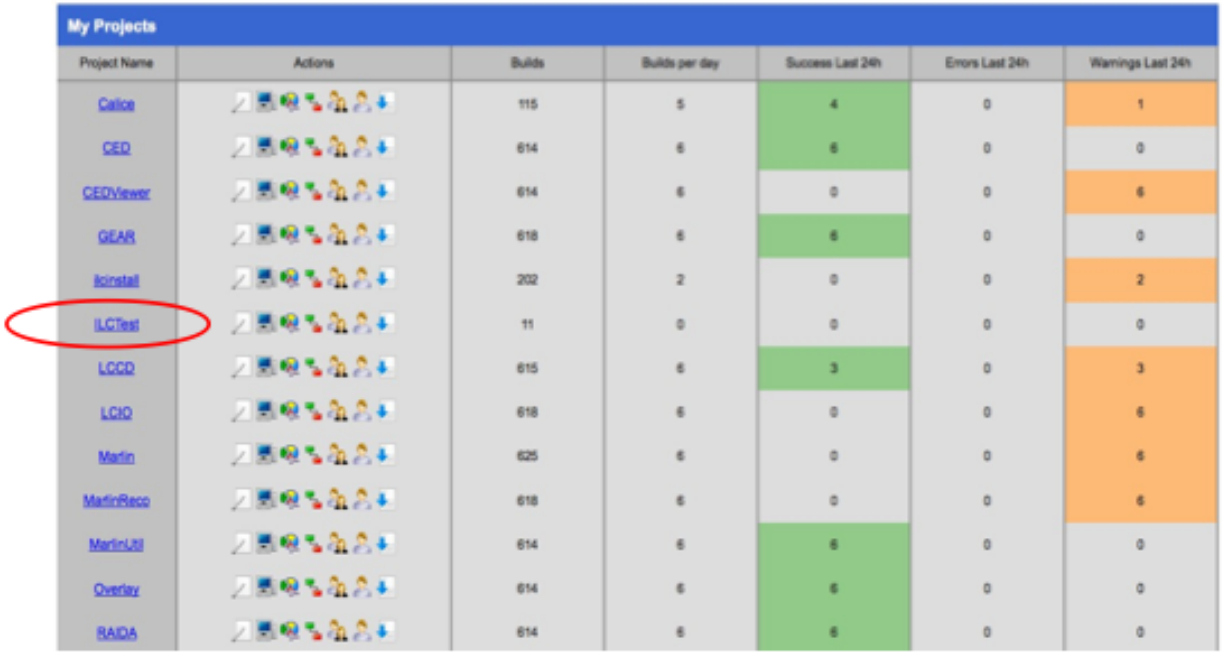


64bit build history

Automatic Email notification if and when build or tests fail.

Developers may subscribe to several projects to keep an overview

iLCTest – package structure



My Projects

Project Name	Actions	Builds	Builds per day	Success Last 24h	Errors Last 24h	Warnings Last 24h
Calice		115	5	4	0	1
CED		614	6	6	0	0
CEDViewer		614	6	0	0	6
GEAR		618	6	6	0	0
lcinstall		202	2	0	0	2
ILCTest		11	0	0	0	0
LCCD		615	6	3	0	3
LCIO		618	6	0	0	6
Marlin		625	6	0	0	6
MarlinReco		618	6	0	0	6
MarlinUtil		614	6	6	0	0
Overlay		614	6	6	0	0
RAIDA		614	6	6	0	0

3 main directories:

doc – Documentation
include – c++ utility headers to simplify writing tests
tests – directory containing the individual tests and cmake macros to simplify adding new tests

Jan Engels

iLCTest – package structure

- C++ header file to simplify writing tests (include/ILCTest.h)
  - // first line in your c++ source file
  - static ILCTest ilctest = ILCTest("hello_world");
    ...
  - ilctest.log("hello world test"); // a log message
    ...
  - If(x != 42){ ilctest.error("wrong answer!!") ; }
    ...
  - cout << last_test_status() << endl; // prints "FAILED"
    ...
  - If(r > 3){ ilctest.fatal_error("this is a fatal error. program will quit now!") ; }
- If all tests were successful, message "TEST_PASSED" is printed. Otherwise "TEST_FAILED". Useful for checking with CTest.

Jan Engels

iLCTest – package structure

- CMake utility macros (tests/CMakeLists.txt)
  - ADD_TEST_DIRECTORY(subdir)
    - cmake macro to add a new test and the corresponding cmake option to turn the test ON or OFF
  - ADD_STD_ILCTEST(testname)
    - cmake macro to add a new c++ test in the current directory and check for the "TEST_PASSED" / "TEST_FAILED" expression given by the utility header include/ILCTest.h (previous slide)

iLCTest – package structure

- tests/simple - directory containing a set of simple tests
  - hello_world – shows a very simple example for writing a c++ test using the ILCTest.h header and the cmake utility macro ADD_STD_ILCTEST
  - fibonnacci – a very simple test which is completely independent of the iLCTest package (just uses python and cmake)
  - root_example – another c++ test which also uses ILCTest.h and additionally uses the ROOT framework to produce and check some dummy histograms
  - marlin_example – a more elaborate test which compiles a marlin plugin, runs Marlin with this plugin and checks for a set of regular expressions in the output of the test.

iLCTest – package structure

```
# 1. initialize ilcsoft
. /afs/desy.de/project/ilcsoft/sw/i386_gcc34_sl4/v01-09-02/init_ilcsoft.sh

# 2. usual cmake build steps
mkdir build
cd build
cmake -C $ILCSOFT/ILCSoft.cmake ..     # to enable ilcsoft tests
make

# 3. run the tests
ctest              # run all tests
ctest -R fibo    # run tests matching regex fibo
ctest -D Experimental    # upload test results to CDash (Experimental)

# (optional) reconfigure build settings
ccmake ..
```

Summary of iLCTest

- Basic Functionality is established
- The fact that the test system has been running for several months will make the release of iLCSoft v01-10 smother than has been the case in the past
- Physics Tests being incorporated
- Unit Testing to be extended