CodeGen User Guide
CDF Note 5641

Dennis Box, Jack Cranshaw, Paolo Calafiura, Rick St. Denis, Jim Kowalkowski

August 14, 2001

Contents

1 Introduction

A C++ Code generator has been implemented to simplify database access for calibration consumers and analysis code for Run II. Users specifiy and generate database tables and C++ accessor code from simple java classes. Usage Examples are presented, along with tables of useful options.

2 Usage Examples

The purpose of this program is to make life easier for the AC++ user by hiding the gory details of adding a new table to the database, and accessing it through an API. An implicit assumption is that a user would prefer to write a simple java class that describes the physical data to be modeled and stored rather than lots of SQL and low level C++ code.

We give examples of:

- adding a simple database table
- accessing this table directly through the Calibration API
- accessing this table directly through an AC++ job
- adding and accessing a database table with byte , int , and float arrays as column types
- accessing a view or a table in a Read-Only fashion (aka Code-Gen Lite)

3 Common Steps

All of the examples in this document follow a general pattern:

- Set up the appropriate environment
- Create a java representation of the table
- Add this java object to a CodeGen dictionary
- Generate the API access and SQL table definition code
- Compile the access code into a library
- Create the database table or view to be accessed
- Use the library to access the database from user code

4 Adding A Simple Database Table

4.1 Set the appropriate environment

Lets assume our unix prompt looks like a '>'. The unix commands:

```
> source ~cdfsoft/cdf2.cshrc
> setup cdfsoft2 development
```

are sufficient to set the environment variables for most uses.

After the environment variables are set, a SoftRelTools[4] development environment must still be created. Lets create a dev area called 'devDir'. It is done like so:

```
>newrel -t development devDir
>cd devDir
>addpkg -h OffCalibDBTables
```

4.2 Create a java representation of the table

Change to the following directory and use your favorite editor to create a file called FakeComponent.java.

```
>cd devDir/OffCalibDBTables/tables
>vi FakeComponent.java
```

The java language description that the CodeGen package uses inherits from emitter.Attrib , a java class in the AACodeGen library. Here is an example java language representation of how a FakeComponent description:

```
import emitter.Attrib;
import emitter.DBString;
import java.io.Serializable;
/**      */
/**                  */
/** Dummy Test table for example purposes    */
public class FakeComponent  extends Attrib implements Serializable
{
    long   dataMember1;
    double dataMember2;
    float  dataMember3;
    static public DBString dataMember4   = new DBString(32);
    static public DBString dataMember5  = new DBString(40);
}
```

4.3 Add your representation to a CodeGen dictionary

In this system, database tables and code objects are logically grouped into 'dictionaries' which may be composed of sub-dictionaries. A closely related term used to group actual physical detector parts and thier calibration data is components, which may be built out of sub-components. A sketch of the entire dictionary and its sub-dictionaries as they existed in July 2000 is shown [3] .

All of the existing sub-dictionaries and table definitions currently live in a *Tables/tables/*comp.java files (see [2]). For instance, new ShowerMax calibration defintions would connect to the ShMaxCalibDBTables/tables/SMXcomp.java sub-dictionary.

We now add the FakeComponent.java file described above, to the dictionary in OffCalibDBTables/tables/TESTcomp.java. Edit the OffCalibDBTables/tables/TESTcomp.java file so it looks like this:

```
import emitter.Attrib;
import emitter.Component;

/**
 *  This is an entry of the CDF calibration dictionary.
 *
 *
 */
 public class TESTcomp extends Component
 {
  public Attrib FakeComponent; //connects FakeComponent to dictionary
  public Attrib TestBytes;
  public Attrib TestFloat;

 }
```

4.4 Generate the API access code and SQL table definition code

The following step should compile your example java program into the dictionary, then read the dictionary and generate C++ and SQL files which you can use in the next step.

- gmake codegen

The codegen step generates C++ and SQL into a temporary directories. The naming convention for these directories works like this: Assume we are working in the 'devDir' directory mentioned earlier. The generated code will go into 'devDir/tmp/(architecture-compiler)', for instance on a Linux box using Kai 4.0 the directory will be 'devDir/tmp/Linux2-KCC_4_0/OffCalibDBTables'. Here is an example of various 'FakeComponent' files that are created during the codegen step:

```
> cd devDir/tmp
> find . -name 'FakeComponent*' -print
./Linux2-KCC_4_0/OffCalibDBTables/OffCalibDBTables_TEXT/FakeComponent.Defs.hh
./Linux2-KCC_4_0/OffCalibDBTables/OffCalibDBTables_TEXT/FakeComponent.hh
./Linux2-KCC_4_0/OffCalibDBTables/OffCalibDBTables_TEXT/FakeComponent.Text.hh
./Linux2-KCC_4_0/OffCalibDBTables/OffCalibDBTables_TEXT/FakeComponent.Text.cc
./Linux2-KCC_4_0/OffCalibDBTables/OffCalibDBTables_MSQL/FakeComponent.Defs.hh
./Linux2-KCC_4_0/OffCalibDBTables/OffCalibDBTables_MSQL/FakeComponent.hh
./Linux2-KCC_4_0/OffCalibDBTables/OffCalibDBTables_MSQL/FakeComponent.pcalib.msql
./Linux2-KCC_4_0/OffCalibDBTables/OffCalibDBTables_MSQL/FakeComponent.Msql.hh
./Linux2-KCC_4_0/OffCalibDBTables/OffCalibDBTables_MSQL/FakeComponent.Msql.cc
./Linux2-KCC_4_0/OffCalibDBTables/OffCalibDBTables_OTL/FakeComponent.Defs.hh
./Linux2-KCC_4_0/OffCalibDBTables/OffCalibDBTables_OTL/FakeComponent.hh
./Linux2-KCC_4_0/OffCalibDBTables/OffCalibDBTables_OTL/FakeComponent.pcalib.osql
./Linux2-KCC_4_0/OffCalibDBTables/OffCalibDBTables_OTL/FakeComponent.OTL.cc
./Linux2-KCC_4_0/OffCalibDBTables/OffCalibDBTables_OTL/FakeComponent.OTL.hh
./Linux2-KCC_4_0/OffCalibDBTables/FakeComponent.class
./Linux2-KCC_4_0/OffCalibDBTables/FakeComponent.Decls.hh
./Linux2-KCC_4_0/OffCalibDBTables/FakeComponent.Defs.hh
./Linux2-KCC_4_0/OffCalibDBTables/FakeComponent.hh
```

Here are brief descriptions of what these files are:

- FakeComponent.hh A C++ class directly generated from FakeComponent.java. It has dataMember1,dataMember2...dataMember5 as private data members, and the following structs, methods and operators:

method	notes
TestFloat()	constructor
TestFloat(data1,data2,....data5)	constructor
TestFloat(struct Raw r)	constructor
dataMember1()	accessor
dataMember2()	accessor
dataMember3()	accessor
dataMember4()	accessor
dataMember5()	accessor
isEqual (FakeComponent & p, float tol=0.01)	equality within tolerance (tol)

operator	notes
==	equality test
!=	inequality test
=	assignment
<	less than
<<	stream input
>>	stream output

- FakeComponent.Defs.hh

  Typedefs for FakeComponentContainer, FakeComponent_mgr and FakeComponentContainer_ptr. A FakeComponentContainer behaves like a std::vector of FakeComponent objects. A FakeComponent_mgr is an object that implements the I/O of the container of FakeComponent objects into the database. A FakeComponentContainer_ptr is a handy pointer to the container.

- FakeComponent.Text.hh ,FakeComponent.Text.cc

  C++ class that implements a manager to perform I/O of containers of FakeComponents to text files.

- FakeComponent.pcalib.msql

  SQL file to create msql table for storing FakeComponent data.

- FakeComponent.Msql.hh FakeComponent.Msql.cc

  C++ class that implements a manager to perform I/O of containers of FakeComponents to an mSQL database table FAKECOMPONENT.

- FakeComponent.pcalib.osql

  SQL file to create oracle table for storing FakeComponent data.

- FakeComponent.OTL.cc, FakeComponent.OTL.hh

  C++ class that implements a manager to perform I/O of containers of FakeComponents to an Oracle database table FAKECOMPONENT.

4.5 Compile the access code into a library

- gmake lib - compiles the previously generated code into libraries. In the example we have been using the libraries would go under 'devDir/lib/Linux2-KCC_4_0'

- gmake nobin - does a gmake codegen followed by a gmake lib

4.6 Create the database table

During 'gmake codegen', we generated 2 scripts in the devDir/tmp area, FakeComponent.pcalib.osql and FakeComponent.pcalib.msql for Oracle and mSQL databases, respectively.

Here is the oracle script:

```
PROMPT Creating Table = FAKECOMPONENT ;
CREATE TABLE FAKECOMPONENT
 (
      CID INTEGER NOT NULL,
      ATTR1 int,
      ATTR2 real,
      ATTR3 real,
      BLA VARCHAR(32),
      ATTR4 VARCHAR(40)
  )

 INITRANS 1
 MAXTRANS 255
 PCTUSED 80
 PCTFREE 10
 STORAGE
 (
 PCTINCREASE 0
 MINEXTENTS 1
 )
 TABLESPACE CALIB_DATA
/

PROMPT Creating Primary Key on FAKECOMPONENT ;
ALTER TABLE FAKECOMPONENT
 ADD CONSTRAINT FAKECOMPONENT_PK PRIMARY KEY
 ( CID, ATTR1 )
 USING INDEX
 PCTFREE 10
 STORAGE
 (
 PCTINCREASE 0
 MINEXTENTS 1
 )
 TABLESPACE CALIB_INDEX
/

PROMPT Creating Foreign Key on FAKECOMPONENT ;
ALTER TABLE  FAKECOMPONENT
  ADD CONSTRAINT FAKECOMPONENT_FK FOREIGN KEY (CID)
  REFERENCES CALIBRUNLISTS(CID);

COMMENT ON  TABLE FAKECOMPONENT IS  ' generated from template
$Id: userguide.tex,v 1.1 2001/07/27 05:16:08 dbox Exp $
with java description
$Id: userguide.tex,v 1.1 2001/07/27 05:16:08 dbox Exp $';

CREATE PUBLIC SYNONYM FAKECOMPONENT  FOR FAKECOMPONENT ;

# -- READ/WRITE_ACCESS are substituted by appropriate role by a script

GRANT SELECT ON FAKECOMPONENT TO CDF_READ;
GRANT SELECT,INSERT,UPDATE ON FAKECOMPONENT TO OffCalib_WRITE;
GRANT SELECT,INSERT,UPDATE ON FAKECOMPONENT TO CALIB_WRITE ;

#
# -- ADD a dummy entry into CALIB_PROPERTIES
#

INSERT INTO CALIB_PROPERTIES
    (CALIB_TABLE,
     CHANNEL_TYPE,
     CHANNELCOUNT,
     FREQUENCY_HOURS,
     WRITE_ROLE,
     CDFNOTE,
     STORAGE_CLASS)
  VALUES (
          'FAKECOMPONENT',
          'FakeComponent',
           10,
           24,
  'OffCalib_WRITE',
   0,
   0.0
          ) ;

PROMPT Finished with Table = FAKECOMPONENT ;
```

This script is intended to create a new table in the Calibration Schema [1]. Enlist the help of an Oracle DBA to perform the following steps:

- Get an oracle account on the database server you intend to develop on
- Log on to the server and run the generated sql scripts to create your table.
- After you have debugged your C++ and SQL code, in the development environment, discuss the procedures to move this into a production environment with your DBA.

5 Use the library to access the database from user code

```
//***********************************************
//File:FakeContainerDriver.cc
//An example program which reads/writes FakeContainer
//This program uses generated files where identified.



//DataKey.hh defines the (calib_run,version) key.
#include "CalibDB/DataKey.hh"

//FakeContainer.Defs.hh has a Data Container and Manager definition

//this is generated code
#include "OffCalibDBTables/FakeContainer.Defs.hh

//FakeContainer.hh has  the Data Object Definition
#include "OffCalibDBTables/FakeContainer.hh

int main(int argc, char* argv[])
{
  //see DBManager/iomap.txt for list of allowable database_ids
  char *database_id ="cdfofdev";
  int  run=1,version=1 ;

  cout << " Program starting ..." << endl;

  // Create a DB manager and a container for table="FakeContainer"
  // @see FakeContainer.Defs.hh
  FakeContainerContainer_ptr gcont;
  FakeContainer_mgr iom(database_id,"FakeContainer");
```

```
  if(iom.isValid()==false)
    {
      cerr << "ERROR: FakeContainer_mgr initialisation failure " << endl;
      return -1;
    }

  // Create a DataKey for run, version
  DataKey qk(run,version);

  //read back from DB
  if(iom.get(qk, gcont)!=Result::success)
    {
      cerr << "ERROR: Failed to retrieve calibration for Run = " << run << endl;
      // Summary
      cout << "read from DB "<< gcont->size()<< " Rows of data " << endl;
    }
  else
    {

     FakeContainerContainer::iterator pi;
     cout << "FakeContainer Contents : Run = " << run
<< " # Rows = " << gcont->size() << endl;
      for(pi=gcont->begin();pi!=gcont->end();++pi)
        dump(cerr, *pi);
    }


  return 0;
}
```

See fillDB.cc and readDB.cc in the DBUtils/Examples/fillDB and DBUtils/Examples/readDB subdirectories for more access examples. These two examples read and write to a 'TestTable' object which in turn writes to an Oracle, Msql, or Text database depending on the input parameters. One can simply replace all instances of 'TestTable' with 'FakeComponent' in these programs and recompile and the programs should work, regardless of how different FakeComponent looks from TestTable.

6 AC++ Access

6.1 AC++ Code

This example accesses the CLAPed calibrations.

```
// Author: Jim Kowalkowski
// Date: 6/2001
// $Id:!CDF_5641_CodeGen_Userguide.ps,v!1.2!2001/08/15!01:57:48!dbox!Exp!$

#include <cassert>
#include <iostream>

#include "DBManager/private/Factory.hh"
#include "ErrorLogger/ErrorLog.h"
#include "Framework/APPUserBuild.hh"
#include "Framework/APPModule.hh"
#include "FrameMods/APPGenInputModule.hh"
#include "FrameMods/CalibrationManager.hh"
#include "CLCCalibDBTables/CLAPed.Defs.hh"
#include "AbsEnv/AbsEnv.hh"

using namespace std;

namespace {

ErrorLog elog("CalibTestMod");

class CalibTestMod : public AppModule
{
public:
  CalibTestMod();
  virtual ~CalibTestMod();

  AppResult beginJob(AbsEvent*);
  AppResult endJob(AbsEvent*);
  AppResult event(AbsEvent*);
  AppResult beginRun(AbsEvent*);
  AppResult endRun(AbsEvent*);

private:
  CLAPed_mgr m_ped_mgr_;
  CLAPed_mgr ped_mgr_;

  CLAPedContainer_ptr m_ped_def_;
  CLAPedContainer_ptr ped_def_;
};
```

```
CalibTestMod::CalibTestMod():
  AppModule("CalibTest","Test the CalibrationManager module")
{

}

AppResult CalibTestMod::beginJob(AbsEvent*)
{
  m_ped_mgr_ = CLAPed_mgr("main","CLAPed");
  ped_mgr_ = CLAPed_mgr("local","CLAPed");

  if(ped_mgr_.isValid()==false)
    {
      elog(ELfatal,"MGR")
<< "local CLAPed manager invalid " << ped_mgr_.lastResult()
<< endmsg;
    }
  if(m_ped_mgr_.isValid()==false)
    {
      elog(ELfatal,"MGR")
<< "main CLAPed manager invalid " << m_ped_mgr_.lastResult()
<< endmsg;
    }

  Result rc;

  if((rc=ped_mgr_.get(ped_def_))!=Result::success)
    {
      elog(ELfatal,"MGRGET")
<< "local CLAPed manager unkeyed get failed " << rc
<< endmsg;
    }
  if((rc=m_ped_mgr_.get(m_ped_def_))!=Result::success)
    {
      elog(ELfatal,"MGRGET")
<< "main CLAPed manager unkeyed get failed " << rc
<< endmsg;
    }

  return AppResult::OK;
}

CalibTestMod::~CalibTestMod( ) { }
```

```
AppResult CalibTestMod::endJob(AbsEvent*)  { return AppResult::OK; }
AppResult CalibTestMod::endRun(AbsEvent*)  { return AppResult::OK; }
AppResult CalibTestMod::beginRun(AbsEvent* e)
{
  cout << "begin run number = " << AbsEnv::instance()->runNumber() << endl;
  return AppResult::OK;
}

AppResult CalibTestMod::event(AbsEvent*)
{
  cout << "run number = " << AbsEnv::instance()->runNumber() << endl;

  cout << "local CLAPed info:\n"
       << " Run=" << ped_def_->run() << "\n"
       << " Version=" << ped_def_->version() << "\n"
       << " Status=" << ped_def_->dataStatus() << "\n"
       << " ID=" << ped_def_->id() << "\n"
       << " size=" << ped_def_->size() << "\n"
       << endl;

  cout << "main CLAPed info:\n"
       << " Run=" << m_ped_def_->run() << "\n"
       << " Version=" << m_ped_def_->version() << "\n"
       << " Status=" << m_ped_def_->dataStatus() << "\n"
       << " ID=" << m_ped_def_->id() << "\n"
       << " size=" << m_ped_def_->size() << "\n"
       << endl;


  return AppResult::OK;
}

}

// ------------------ user build stuff -------------------

AppUserBuild::AppUserBuild(AppFramework* f):AppBuild(f)
{
  add( new APPGenInputModule );
  add( new CalibrationManager );
  add( new CalibTestMod );
}

AppUserBuild::~AppUserBuild( ) { }
const char * AppUserBuild::rcsId( ) const { return "$Id"; }
```

6.2 Tcl Driver for AC++ Code

This tcl file drives the above CLAPed example to access run number 119444 from the online production database.

```
path enable AllPath
talk CalibrationManager
  add main  OTL "cdf_reader/reader@cdfofprd"
  add local  OTL "cdf_reader/reader@cdfofprd"
  # add local Text testdb

  # add main  Text testdb
  # add local OTL  "cdf_reader/reader@cdfonprd"

  Database set main
  DataDB   set main

  Debug    set true
  ProcessName set PROD_PHYSICS_CDF
exit

talk GenInputModule
  events set 1
  run_number set 119444
exit

begin -ev 1
exit
```

7 Byte, Int, and Float Arrays

For some applications it is convenient to model the data as objects containing arrays of bytes, integers or floating point data. This is conveniently handled in the CodeGen system. The following java description file has examples of all three array types:

```
import emitter.Attrib;
import emitter.DBString;
import java.io.Serializable;

public class TestBytes  extends Attrib implements Serializable
{
    long   attr1;
    public byte[] ped1 = new byte[128];
    public float[] ped2 = new float[128];
```

```
    public int[] ped3 = new int[128];
}
```

8 Read-Only Access With Code Gen Lite

If the java description files used in the previous examples are made to inherit from the 'ROattrib' class instead of 'Attrib', the resulting C++ code can perform very flexible SELECT's, but no INSERT's on its database table. The SELECT statement is modified by means of an ROKey, which takes a WHERE clause as its only argument, and appends it to the default select statment for the table. . See the DBViews package, or the DBUtils/Examples/readView subdirectory for working examples.

Read-only access to database tables outside the of Calibration DB is possible by proper subclassing of the java description files.

9 Environment Variables

Environment variables set by these commands that may be of interest to a modifier of the code generator include:

- OMIT_MSQL - turns of the generation, compiling, and linking of MSQL database access code
- OMIT_ORACLE - does the same for oracle code
- CDF_ORACLE_CLIENT_EXISTS - an older 'switch' that fulfilled the same purpose as OMIT_ORACLE does now. The reason it is mentioned here is that it still may lurk about in makefiles, and if you set OMIT_ORACLE to 1 you should also unset CDF_ORACLE_CLIENT_EXISTS for safety (or fix the makefiles (!)).
- OMIT_TEXTDB - does the same for text databases

10 Java to C++ and SQL Conversions

Java Primitive or Class	C++ Class	Oracle SQL Type	Notes
java.lang.long	long	int	
java.lang.double	double	float	
java.lang.float	float	float	
java.lang.int	long	int	
java.lang.char	char	TEXT(1)	
array of java.lang.char	char *	TEXT(N)	
emitter.DBString	std::string	VARCHAR(N)	
array of java.lang.int	std::vector of int	LONG RAW(N)	
array of java.lang.float	std::vector of float	LONG RAW(N)	
array of java.lang.byte	std::vector of unsigned	LONG RAW(N)	

10.1 Planned But Not Implemented (Yet)

Java Primitive or Class	C++ Class	Oracle SQL Type	Notes
java.util.Date	???	DATE	
???	long long	RAW	

10.2 A Java Example

```
public class AnotherFakeComponent  extends Attrib implements Serializable
{


    long   aLong;
    double aDouble;
    float  aFloat;
    int    anInt;
    char   aChar;
    static public char aCharArray[] = new char[72];
    static public DBString aDBString   = new DBString(32);
    static public int  anIntArray[] = new int[12];
    static public float aFloatArray[] = new float[245];
    static public byte  aByteArray[] = new byte[256];

};
```

10.3 A C++ Snippet Generated from Java Example

```
class AnotherFakeComponent
{
public:
//.lots of stuff deleted for brevity
private:
```

```
   long _aLong;
   double _aDouble;
   float _aFloat;
   long _anInt;
   char _aChar;
   char* _aCharArray;
   std::string _aDBString;
   std::vector<int> _anIntArray;
   std::vector<float> _aFloatArray;
   std::vector<unsigned char> _aByteArray;

};
```

10.4 Oracle SQL Snippet Generated from Java Example

```
CREATE TABLE ANOTHERFAKECOMPONENT
  (
      CID INTEGER NOT NULL,
      ALONG int,
      ADOUBLE real,
      AFLOAT real,
      ANINT int,
      ACHAR text,
      ACHARARRAY text(72),
      ADBSTRING VARCHAR(32),
      ANINTARRAY RAW(48),
      AFLOATARRAY RAW(980),
      ABYTEARRAY RAW(256)

  )
```

References

[1] "A Calibration Database Schema for CDF II," Jack Cranshaw, Mark Lancaster, Jim Kowalkowski, Paolo Calafiura, Marjorie Shapiro, Dennis Box, Robert Harris. CDF Note 5088, 8/12/99 revised 2/12/01

[2] "CVS source tree for AACodeGen package, ", The CDF Collaboration. http://cdfcodebrowser.fnal.gov/CdfCode/

[3] "CodeGen Design Notes and Status," Dennis Box, Paolo Calafiura, Jim Kowalkowski. CDF Note 5460 ,9/25/96 revised 6/24/99

[4] "A UNIX based Software Management System," Robert Harris. CDF Note 3891,9/25/96 revised 6/24/99

[5] "Design Patterns Elements of Reusable Object Oriented Software," Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Addison-Wesley Publishing Co. 1995 ISBN 0-201-63361-2 pp. 331-344

[6] "Creating Talkto Menus for AC++," Elizabeth Sexton-Kennedy, Marjorie Shapiro CDF Note Number: CDF/DOC/COMP_UPG/PUBLIC/4521 Date: 3/5/98

[7] "A Python-based prototype for AC++ user interface," P.Murat CDF Note Number: CDF/DOC/COMP_UPG/CDFR/4344 Date: 9/25/97

[8] "A Users Guide to the AC++ Framework," Elizabeth Sexton-Kennedy CDF Note Number: CDF/DOC/COMP_UPG/PUBLIC/4178 Date: 5/16/97