May 2000 Prototype Framework Tutorial: Examples

Paolo Calafiura, Charles Leggett

HCG/NERSC/LBNL

ATLAS Software WorkShop @ LBNL

May 9, 2000

Example 1:
HelloWorld

- **Concepts**

- **Part 0:**
  - **set up build environments**
  - **create a new package**
  - **compile and link**
  - **run**
- **Part 1:**
  - **The Messaging Service**
- **Part 2:**
  - **The JobOptions and Properties Service**

- **Files:**

  - **HelloWorld.cxx, HelloWorld.h**

Example 1: Part 0
Setup Build Environments

```
> source /auto/atlas/tools/WorkshopSetup.csh
```

- Work area: `~/Workshop`

```
> cd ~/Workshop
```

- source files are in

  `src41/ControlExamples/GaudiTutorial/src`

- header files in

  `src41/ControlExamples/GaudiTutorial/GaudiTutorial`

- build from `~/Workshop/build41`

```
> gmake install
```

- execute from `~/Workshop/run`

```
> ./tutorial_examples < atlas.datback > output
```

- solutions to examples in

  `~/Workshop/solutions/ex1/part1 ...`

Example 1: Part 0
Initial Files

- For each example, copy initial files from
  `~/Workshop/solutions/ex(n)`
- copy source files (*.cxx) to your source file area.

```
> cd ~/Workshop/src41/ControlExamples/GaudiTutorial/src
> cp ~/Workshop/solutions/ex1/*.cxx .
```

- copy header files (*.h) to your header file area.

```
> cd ../GaudiTutorial
> cp ~/Workshop/solutions/ex1/*.h .
```

- copy GNUmakefile.in

```
> cd ..
> cp ~/Workshop/solutions/ex1/GNUmakefile .
```

- copy jobOptions.txt to run area.

```
> cd ~/Workshop/run
> cp ~/Workshop/solutions/ex1/jobOptions.txt .
```

Example 1: Part 0
Shortcuts

- **Aliases have been defined to cd to various directories:**

```
gotosrc == cd
  ~/Workshop/src41/ControlExamples/GaudiTutorial/src
gotohdr == cd
  ~/Workshop/src41/ControlExamples/GaudiTutorial/GaudiTutorial
gotobld == cd ~/Workshop/build41
gotorun == cd ~/Workshop/run
gotosol == cd ~/Workshop/solutions
```

- **Initialization files can be automatically copied to the correct directories with:**

```
WorkshopInit example#
```

- **The full solutions can be copied to the correct directories with:**

```
WorkshopCheat example# part#
```

- **If you do this, don't forget to "gmake build" afterwards**
- **To filter out SLUG output:**

```
./tutorial_examples < atlas.datback | filter
```

Example 1: Part 1
The Messaging Service

- **Add Message Service header to HelloWorld source:**

```
#include "Gaudi/MessageSvc/MsgStream.h"
```

- **To print to message service:**

```
MsgStream log(messageService(), name());
log << MSG::INFO << "hello world" << endreq;
```

- **Various message levels and jobOptions equivalents:**

```
MSG::DEBUG      == 2    DEFAULT
MSG::INFO       == 3
MSG::WARNING    == 4
MSG::ERROR      == 5
MSG::FATAL      == 6
```

Example 1: Part 1
Messaging Service (cont)

- **add Message Service output level to jobOptions.txt**

```
MsgService.OutputLevel = 2;
```

- **Can set output level for each algorithm independently**

```
HelloWorld.OutputLevel = 4;
```

- **To tell the framework to run the algorithm, add the Algorithm name to the jobOptions file:**

```
ApplicationMgr.TopAlgorithm = {“HelloWorld”};
```

- **Now edit the files, build the executable, and run it.**

Example 1: Part 1
Accessing Services from Algorithms

- **Any registered service can be accessed from an algorithm once its header file has been included in the algorithm.**
- **The algorithm class provides hooks to the various services.**
- **The most common services are:**

```
messageService()
eventDataService()
histogramDataService()
ntupleService()
detDataService()
serviceLocator()
and more....
```

Example 1: Part 2
Properties

- Algorithms can have properties, which are settable at run time via the jobOptions file.
- In the source file of HelloWorld, include the the header for the Property Manager:

```
#include "Gaudi/JobOptionsSvc/PropertyMgr.h"
```

- Declare the property variable in header file as a private data member:

```
int m_foo;
```

- In Algorithm constructor, declare the property:

```
declareProperty("Foo", m_foo);
```

- This makes an association between a data member and a property name as used in the jobOptions
- The supported data types are:

```
int, double, bool, string
std::vector<int>, std::vector<double>,
  std::vector<bool>, std::vector<string>
```

Example 1: Part 2
Properties (cont)

- **Add the property entry in the jobOptions file:**

```
HelloWorld.Foo = 42;
```

- **Do the same thing for a bool, a double, and a string vector.**
- **For vectors the syntax is:**

```
HelloWorld.StringVecFoo = {“Hello”, “World”};
```

- **Print out the value of the properties using the messaging service in the execute() method.**

Example 1: Postscript
SimpleProperties

- A new type of Properties class called a "SimpleProperty" has been added to the framework, which provides the ability to do bounds checking. Though not part of this tutorial, it will become the preferred usage of properties.
- The properties member data will be declared as:

```
SimpleProperty<int> m_int;
```

  - or

```
IntegerProperty m_int;
```

- Optional bounds can be set in the initialize() method as:

```
m_property.setBounds(lower, upper);
m_property.setUpper(upper);
m_property.setLower(lower);
```

Example 2: Transient Data Store, Multiple Algorithms

- **Concepts:**

- **Part 1:**
  - **Getting to the Transient Data Store**
  - **Getting the event header from the TDS, and accessing its components**
- **Part 2:**
  - **Creating a DataObject**
  - **Writing DataObject to TDS**
  - **Reading DataObject from TDS**
  - **Chaining multiple algorithms**

Example 2: Transient Data Store, Multiple Algorithms

- **Part 3:**
  - **Creating a ContainedObject**
  - **Writing a vector of ContainedObjects to TDS**
  - **Reading vector of ContainedObjects from TDS**

Files:

- **ReadData.cxx, ReadData.h**
- **WriteData.cxx, WriteData.h**
- **DataObj.h, ContObj.h**

Example 2:
Transient Data Store

- Event information is kept in the TDS under the /Event hierarchy.
- There are several ways to get objects out of the TDS. These include the findObject() and retrieveObject() methods from the IDataProviderSvc, as well as the SmartDataPtr<>, and SmartRef<> templates.
- We prefer to use the SmartDataPtr as it provides the cleanest interface.
- When an object is registered with the TDS, the TDS becomes the owner of the object. DON'T CALL THE DELETE METHOD OF THE OBJECT.
- See chapter 6 of the Gaudi manual for further details.

Example 2: Part 1
Getting to the TDS

- Start with the ReadData algorithm skeleton
- Add the headers for the ZebraTDREvent, the event data service, and the Smart Data Pointer:

```
#include "ZebraTDRCnv/ZebraTDREvent.h"
#include "Gaudi/DataSvc/EvtDataSvc.h"
#include "Gaudi/DataSvc/SmartDataPtr.h"
```

- In the execute() method of ReadData, Get a SmartDataPtr to the Event in the TDS:

```
SmartDataPtr<ZebraTDREvent>
  evt(eventDataService(),"/Event");
```

- Check to see if a valid event was found:

```
if (!evt) { print error message, return FAILURE }
```

- Get the event and run numbers:

```
int event = evt->event_number();
int run = evt->run_number();
```

Example 2: Part 2
Creating a DataObject

- **To be registered in the TDS, an object must be a DataObject, ie it must inherit from DataObject.**
- **You must also define a static constant Class ID, which must be unique for each class. In MyDataObj.h:**

```
#include "Gaudi/Kernel/DataObject.h"
const static CLID CLID_MDO = 214222;
class MyDataObj: public DataObject {
 public:
   MyDataObject(): DataObject(){};
   static CLID& classID() { return CLID_MDO; }
   virtual CLID& clID() { return CLID_MDO; }
   int val() { return m_dat; }
   void val( int i) { m_dat = i; }
   virtual ~MyDataObj(){};
 private:
   int m_dat;
};
```

Example 2: Part 2
Writing DataObject to TDS

- **The DataObject will be written to a location in the TDS as specified by a property in jobOptions:**

```
declareProperty("DataObjLoc", m_DataObjLoc);
```

- **Add this to both the ReadData and WriteData algorithms, and in the jobOptions file.**
- **In the execute() method of WriteData, create a DataObject of type MyDataObj, and set its member data:**

```
MyDataObj *dobj = new MyDataObj;
dobj->val(10);
```

- **Register it in the event TDS**

```
StatusCode sc = eventDataService()->
  registerObject(m_DataObjLoc,dobj);
if ( sc.isFailure(){ print error message }
```

Example 2: Part 2
Read DataObject from TDS

- **Objects in the TDS can be accessed via a SmartDataPtr<>, which is templated in the type of the object, and has a usage syntax similar to a normal pointer.**
- **In the execute() method of ReadData, get a SmartDataPtr of type MyDataObj from the event TDS, with its location specified by the property variable:**

```
SmartDataPtr<MyDataObj>
  dobj(eventDataService(),m_DataObjLoc);
if (! dobj) { print error message }
```

- **Print value of object member data:**

```
log << MSG::INFO << "Data Object Val: "
    << dobj->val() << endreq;
```

Example 2: Part 2
DataObject Locations

- **In the jobOptions file, specify the locations of the objects to be written / read:**

```
WriteData.DataObjLoc = "/Event/tmpD";
ReadData.DataObjLoc = "/Event/tmpD";
```

- **Make sure that you're reading from the same location as you're writing to**

Example 2: Part 2
Chaining Multiple Algorithms

- In order to execute more than one algorithm, all that must be modified is the “TopAlg” property of the Application Manager in the jobOptions file:

```
ApplicationMgr.TopAlg = { “WriteData”,
  “ReadData”} ;
```

- Modify the GNUmakefile.in so that WriteData.cxx gets compiled:

```
> cd ~/Workshop/src41/ControlExamples/GaudiTutorial
libGaudiTutorial.so_SRC = src/ReadData.cxx
  src/WriteData.cxx
```

- Compile and link using “gmake install”
- run the executable in the run directory

Example 2: Part 3
Creating a Contained Object

- **If you want to store a collection of objects, such as a group of hits, in the TDS, you should use an ObjectVector or an ObjectList.**
- **Objects must inherit from ContainedObject if they are to be stored in an ObjectVector/List:**

```
#include "Gaudi/Kernel/ContainedObject.h"
const static CLID CLID_MYCO = 214223;
class MyContObject: public ContainedObject {
 public:
   MyContObject(): ContainedObject(){};
   static CLID& classID() { return CLID_MYCO; }
   virtual CLID& clID() { return CLID_MYCO; }
   void set(float t, int i) { m_time = t; m_channelID= i;}
   int id() { return m_channelId; }
.....
 private:
  int m_channelId;
  float m_time;
};
```

Example 2: Part 3
Writing a Contained Object

- **In the execute() method of WriteData, create and fill a ContainedObject with the hits:**

```
MyContObj *mco_1 = new MyContObj;
mco_1->Set(11.3, 1);
....
```

- **Create an ObjectVector on the heap, and fill it with a contained object:**

```
ObjectVector<MyContObj> *cobj = new
  ObjectVector<MyContObj>;
cobj->push_back(mco_1);
cobj->push_back(mco_2);
...
```

Example 2: Part 3
Writing a Contained Object

- **Register the ObjectVector with the TDS, and check to see that all went well:**

```
sc = eventDataService()->
  registerObject(m_ContObjLoc, cobj);
if ( sc.isFailure() ) { print error mesg };
```

- **In the jobOptions file, specify the location of the object to be written:**

```
WriteData.ContObjLoc = "/Event/tmpC";
ReadData.ContObjLoc = "/Event/tmpC";
```

- **Make sure it's a different location that what was specified for the DataObject. Only one object can be registered to any one location.**

Example 2: Part 3
Reading Contained Object from TDS

- **In the execute() method of ReadData, get a SmartDataPtr that points to the contained object in the TDS:**

```
SmartDataPtr< ObjectVector<MyContObj> >
  vec(eventDataService(), m_ContObjLoc);
```

- **Iterate through the vector, printing out the values of its members:**

```
ObjectVector<MyContObj>::iterator it;
for (it=vec->begin(); it!=vec->end; it++) {
  int Id = (*it)->id();
  float time  = (*it)->time();
  log << MSG::INFO << "ID: " << Id << " Time: "
      << time << endreq;
}
```

- **Now build and run the executable**

Example 3
Sub-Algorithms

- **Concepts:**

- **Part 1:**
  - **Using sub algorithms**
- **Part 2:**
  - **Using multiple identical sub-algorithms**

- **Files**

- **MainAlg.cxx, MainAlg.h**
- **SubAlg.cxx, SubAlg.h**

Example 3: Part 1
Sub-Algorithms

- **Sub-algorithms allow a higher degree of user control over the execution and ordering of algorithms.**
- **Sub-algorithms are not automatically created by the framework - this has to be done explicitly by their parent algorithm.**
- **Once created, the initialize() method of the sub-algorithm is automatically called, but the execute() phase must be explicitly invoked by the parent algorithm.**
- **The finalize() method is automatically called by the framework.**

Example 3: Part 1
Sub-Algorithms

- **The sub-algorithm (SubAlg) will be very simple, with just a single string property called "SubString"**

```
declareProperty("SubString", m_substr);
```

- **In the initialize() and execute() methods of the sub-algorithm, we want to print out the value of its string property.**

Example 3: Part 1
Sub-Algorithms

- **The sub-algorithm will be referenced via a string property of the parent algorithm (MainAlg).**

```
declareProperty("SubAlgType",m_subAlgType);
declareProperty("SubAlgName",m_subAlgName);
```

- **The properties of the sub-algorithm will also be set by properties of the parent algorithm**

```
declareProperty("SubAlgPropName",
                m_subAlgPropName);
declareProperty("SubAlgPropVal",m_subAlgPropVal);
```

- **In order to identify the sub-algorithm, we want to keep a pointer to it as a member data of the parent algorithm. In the header of MainAlg:**

```
Algorithm* p_Sub;
```

Example 3: Part 1
Sub-Algorithms

- **In the initialize() method of the parent algorithm, the sub-algorithm must be created:**

```
sc = createSubAlgorithm(m_subAlgType,
                        m_subAlgName,p_Sub);
if ( sc.isFailure() ) { print error message }
```

- **In the execute() method of the parent algorithm, we want to set a property of the sub-algorithm:**

```
sc = p_Sub->setProperty( StringProperty
  (m_subAlgPropName, m_subAlgPropVal) );
```

- **Then we want to execute the sub-algorithm:**

```
sc = p_Sub->execute();
```

Example 3: Part 2
Multiple Identical Sub-Algorithms

- **In order to execute multiple identical sub-algorithms, each one must be created by the parent algorithm with a different name.**
- **First change the SubAlgName and SubAlgPropVal properties in the parent algorithm to be vector<string>. Same for p_Sub which will become a vector<Algorithm*>.**
- **Then in the initialize() method of the parent alg:**

```
m_nSubAlg = m_subAlgName.size();
Algorithm* psb;
for (int i=0; i< m_nSubAlg; i++) {
   sc = createSubAlgorithm(m_subAlgType, m_subAlgName[i],
        psb);
   if ( sc.isFailure() ) { print error message }
   p_Sub.push_back(psb);
}
```

Example 3: Part 2
Multiple Identical Sub-Algorithms

- **In the execute() method, loop over all sub-algorithms, setting and printing their properties**

```
for (int i=0; i<m_nSubAlg; i++) {
   log << MSG::INFO << "setting property "
       << m_subAlgPropName << " to "
       << m_subAlgPropVal[i] << " for subalg "
       << m_subAlgType << " instance "
       << m_subAlgName[i] << endreq;

   sc = p_Sub[i]->setProperty( StringProperty
        (m_subAlgPropName, m_subAlgPropVal[i]) );
   if ( sc.isFailure() ) { print error message }

   sc = p_Sub[i]->execute();
}
```

Example 3: Part 2
Multiple Identical Sub-Algorithms

- **In the jobOptions file:**

```
MainAlg.SubAlgType = "SubAlg";
MainAlg.SubAlgPropName = "SubString";
MainAlg.SubAlgName = { "SubAlgInst_1",
        "SubAlgInst_2", "SubAlgInst_3" };
MainAlg.SubAlgPropVal = { "new string 1",
        "new string 2", "new string 3" };

SubAlgInst_1.SubString = "original string 1";
SubAlgInst_2.SubString = "original string 2";
SubAlgInst_3.SubString = "original string 3";
```

Example 3: Postscript
Sequencer

- A new algorithm type called a sequencer has been added to the framework. The sequencer provides the same functionality as the sub-algorithms, but in a more automatic manner.
- Instead of explicitly creating and executing the sequencer in the appropriate methods of the parent algorithm, the sequencing path is set via the jobOptions:

```
ApplicationMgr.TopAlg = { “Sequencer/TopSequencer” };
TopSequencer.Members={“Sequencer/Path1”,“Sequencer/Path2”};
Path1.Members = {“hello”,”Prescaler/Prescaler1”,”goodbye”};
Path2.Members = {“hello”,”Prescaler/Prescaler2”,”goodbye”};

Prescaler1.percentPass = 20.0;
Prescaler2.percentPass = 50.0;
```

Example 4
Histograms and NTuples

- **Concepts**

- **Part 1:**
  - **Histograms**
- **Part 2:**
  - **Ntuples**

- **Files**

- **Hist.cxx, Hist.h**
- **Ntup.cxx, Ntup.h**

Example 4: Part 1
Histograms

- Histograms are kept in the TDS in a special area, which, unlike the event store, is not cleared with each new event.
- This area is “/stat”, and is managed by the histogram service HistogramSvc.
- Histograms must be registered with the TDS, like any other data object.
- The framework uses HTL histograms, and saves them in the hbook format.

Example 4: Part 1
Histograms

- **In the header file, add the data member:**

```
IHistogram1D* m_hist;
```

- **In the source file, add the headers:**

```
#include "Gaudi/Interfaces/IHistogramSvc.h"
#include "Gaudi/HistogramSvc/H1D.h"
```

- **In the initialize() method, create a histogram object, and register it:**

```
m_hist = new H1D("TestHist", "Test Histogram",
         "1",100,0,100);
sc = histogramDataService() -> registerObject(
     "/stat/simple"+m_hist->number(),m_hist);
if ( sc.isFailure() ) { print error message }
```

Example 4: Part 1
Histograms

- **In the execute() method, fill the histogram:**

```
m_hist->fill( float(event), 1.);
```

- **In the jobOptions file, specify the name of the output file:**

```
HistogramPersistencySvc.OutputFile =
  “histo.hbook”;
```

- **Compile and execute**
- **Run PAW to look at the output**

Example 4: Part 2
NTuples

- NTuples are managed by a service similar to the histogramming service - NTupleSvc. They are kept in the TDS in their own area, called "/NTUPLES".
- The framework saves the NTuples in hbook format.
- Both column-wise and row-wise NTuples are supported.

Example 4: Part 2
NTuples

- **For a column-wise NTuple, the variables must first be declared as data members of the class:**

```
NTuple::Item<long> m_size;
NTuple::Item<long> m_run,m_event;
NTuple::Array<long> m_rundata;

NTuple::Tuple* p_nt1;
```

- **In the source file, add the includes:**

```
#include “Gaudi/Interfaces/INTupleSvc.h”
#include “Gaudi/NTupleSvc/NTuple.h”
#include “Gaudi/NTupleSvc/NTupleDirectory.h”
```

Example 4: Part 2
NTuples

- **In the initialize() method, create and register the ntuple:**

```
NTuple::Directory *col=0;
NTupleFilePtr file1(ntupleService(),
                    "/NTUPLES/FILE1");
if ( !file1 ) {
  if ( !( col=ntupleService()->
     createDirectory("/NTUPLES/FILE1/COL") ){
       print error message: can't create dir
  }
} else {
   print error message: can't create file
}
```

Example 4: Part 2
NTuples

```
NtuplePtr nt1(ntupleService(),
  "/NTUPLES/FILE1/COL/10");
nt1= ntupleService()-> book(
     col,10,CLID_ColumnWiseTuple, "Col Wise");
if ( nt1 ) {
   p_nt1 = nt1;
   sc = nt1->addItem("Event", m_event);
   if ( sc.isFailure() ) { print error }
   sc = nt1->addItem("Run", m_run);
   sc = nt1->addItem("Size", m_size, 0, 100);
   sc = nt1->addItem("rundata", m_size,
        m_rundata);
} else {
   print error: can't book ntuple
}
```

Example 4: Part 2
NTuples

- **In the execute() method, fill the ntuple**

```
m_size = 2 + (evt->event_number() %3);
m_event = evt->event_number();
m_run = evt->run_number();
m_rundata[0] = m_run;
m_rundata[1] = m_event;
for (int i=2; i<m_size; i++) {
   m_rundata[i] = i;
}

sc = ntupleService()->writeRecord(p_nt1);
if ( sc.isFailure() ) { print error message }
```

Example 4: Part 2
NTuples

- **In the jobOptions file, set the output file name and the file format type (only hbook works for now)**

```
NTupleSvc.Output = {
  “FILE1#tuple1.hbook(WRITE)” };

NTupleSvc.Type = 6;
```

- **run it and look at the ntuple written to the file “tuple1.hbook” with PAW.**