Algorithm 101:
Printing and Configuration

Writing Your Own Algorithms

Practically all physics-related tasks are implemented in Athena as Algorithms

- **Simulation: GeneratorModules/SingleParticleGun**
- **Reconstruction:**
- **Analysis: RecExample/ZeeBuilder**

An Athena Algorithm is a C++ class that implements the GaudiKernel IAlgorithm interface

- **this is usually accomplished by inheriting from GaudiKernel Algorithm base class.**

ZeeBuilder.h

```
class ZeeBuilder : public Algorithm {
 public:
  ///standard gaudi algorithms constructor
  ZeeBuilder(const std::string& name,
             ISvcLocator* pSvcLocator);
  ~ZeeBuilder();

  /// \name IAlgorithm implementation
  //@{
  StatusCode initialize();  //< run once at begin of job
  StatusCode execute();     //< run once per event
  StatusCode finalize();    //< run once at end of job
  //@}
  ...
};
```

Accessing Services

Within the Algorithm services are readily accessible.

The most commonly used accessors are:

- **msgSvc( )** [or messageService( )]
- **histoSvc( )** [or histogramDataService( )]
- **ntupleSvc( )** [or ntupleService( )]
- **service<T>(…)** generalized access to Services
- **serviceLocator( )**

And more…

For historical reasons, the Algorithm base class does not provide an accessor to StoreGateSvc (the TDS client API)

Message Service

Allow to log text messages, with different level of detail, and an identifier of the issuer.

Why not use std::cout, std::cerr, ... ?

- **They surely print, but**
  - **Do you always want to print to the log file?**
  - **How can you connect std::cout to the message window of an event display?**
  - **You may want to switch on/off printing at several levels just for one given algorithm, service etc.**

Message Service

Use of the MsgStream class

- **Usable (almost) like standard C++ streams**
- **Different levels of printing - OutputLevel:**

MSG::NIL (=0) DEFAULT
MSG::VERBOSE (=1)
MSG::DEBUG (=2)
MSG::INFO (=3)
MSG::WARNING (=4)
MSG::ERROR (=5)
MSG::FATAL (=6)
MSG::ALWAYS (=7)

- **Record, not line oriented**
- **Allows to define the severity level per object instance**

```
MessageSvc.OutputLevel = 3;     //default level INFO
MyAlgorithm.OutputLevel = 2;  //MyAlgorithm level is DEBUG
MessageSvc.setVerbose += {“MyObject”};  //MyObject is VERBOSE
```

MsgStream: Example of use

Print messages with MsgStream

Include message stream header

- *#include "GaudiKernel/MsgStream.h"*

Create MsgStream object and print

For example, in `initialize()`

```
MsgStream log(msgSvc(), name());
log  << MSG::INFO
     << "Initializing ZeeBuilder"
     << endreq;
```

To avoid costly computation you can also query the message level

```
if(msgSvc()->outputLevel(name()) <= MSG::DEBUG) {
    log << streamingBigTime() << endreq;
}
```

Hands On: more ASK

Use ask to checkout

Reconstruction/RecExample/ZeeRec (use tag 6.5.0)

Rebuild (using broadcast) RecExCommon

Go to ZeeRec/ZeeRec-00-02-01/src

Run script ../share/swapfile.sh

Go to RecExCommon/RecExCommon-00-01-12/run

Edit RecExCommon_jobOptions.txt, and add

```
// to load and execute ZeeRec algos
#include "$ZEERECROOT/share/ZeeRec_jobOptions.txt"
```

And Finally, type from the run directory

```
magda_getfile dc1.002046.simul.00001.hlt.Z_ee.zebra
ln -sf dc1.002046.simul.00001.hlt.Z_ee.zebra ZEBRA.P
```

Hands On: MsgStream

Print messages with MsgStream

Add message stream header to ZeeBuilder.cxx

```
#include "GaudiKernel/MsgStream.h"
```

Create MsgStream object and print

For example, in ZeeBuilder::initialize()

```
MsgStream log(msgSvc(), name());
log << MSG::INFO << "Initializing ZeeBuilder" << endreq;
```

Athena job configuration

Job is essentially steered by a conventional text file or a Python script

Options & properties are accessed through framework interfaces, IJobOptionsSvc or IScriptingSvc

JobOptions details

Common entries in jobOptions.txt

Example of Standard Configuration

```
#include "Atlas_Gen.UnixStandardJob.txt”
#include “StandardAtlfastOptions.txt”
```

Maximum number of events to execute

```
ApplicationMgr.EvtMax   = <integer>
```

Component libraries to be loaded

```
ApplicationMgr.DLLs += {<comma separated array of string>}
```

Top level algorithms: “Type/ObjectName”

```
ApplicationMgr.TopAlg += {<comma separated Array of string>}
```

Comments

Preceded by //

Job Options: Data Types

Primitives

bool, int, double, long, std::string

Arrays of primitives

std::vector<bool>, std::vector<double>, ...

More...

- See Gaudi User's Guide

Hands On: Declaring a Property

Add a couple of properties to ZeeBuilder

- In ZeeBuilder.h

  declare properties as private data member

- In ZeeBuilder.cxx

  call Algorithm::declareProperty method that associates to each property

  - Its job options name (“eGammaMin”)
  - (optionally) a default value (2)

ZeeBuilder.h

```
class ZeeBuilder : public Algorithm {
  ...
private:

  ...
  std::string m_egContainerName;  //< SG key
  //Selections
  double m_electronPtMin; //< Electron PtMin cut
  int egammaMin;   //< minimum number of egammas
  ...
};
```

ZeeBuilder.cxx

```
ZeeBuilder::ZeeBuilder(const std::string& name,
                       ISvcLocator* pSvcLocator):
  Algorithm(name, pSvcLocator)
{
  ...
  // container keys
  declareProperty("egContainer",m_egContainerName);
  declareProperty("ZeeContainer",m_ZeeContainerName);

  // Selections
  declareProperty("electronPtMin",m_electronPtMin=10.0);
  declareProperty("eGammaMin",m_eGammaMin=1);
  ...
}
```

Hands On: Set Properties

Set property in ZeeRec_jobOptions.txt

ZeeBuilder.eGammaMin = 2;

Print property values and verify they have changed

- *in ZeeBuilder::initialize*

```
log    << MSG::DEBUG
       << "Minimum number of egammas"
       << m_eGammaMin
    << endreq;
```

Hands On: Use Colors

Remember to uncomment property in run/RecExCommon_jobOptions.txt

MessageSvc.useColors = true;

- ***This allows to use colorized messages***

Now tell MsgStream to print your msgs in red

- ***For example in ZeeBuilder::initialize***

```
log.setColor(MSG::RED);
// or log.setColor(MSG::YELLOW, MSG::BLUE);
log    << MSG::DEBUG
       << "Minimum number of egammas"
       << m_eGammaMin
    << endreq;
```

Use less -r to parse run/AskAthena.log