
Overview of ATLAS Software

and the Athena Framework

(extracted from an earlier talk by S. R. & D. R.)



Overview of the ATLAS Software

- ❖ General introduction to ATLAS software
- ❖ CVS, Packages, build tools and Releases
- ❖ Introduction to Athena Software
 - Algorithms, Tools, Services
 - Configuring jobs via jobOptions



CVS, CMT, Releases ... ??

- ❖ All atlas offline code is stored in **CVS**, which manages the evolution of the source files
- ❖ The build of the binaries (compilation option, include files, libraries...) as well as the run-time environment are managed by **CMT**
- ❖ A new version of the code is entirely rebuilt approximately every 3 weeks (« developer release » e.g 8.7.0) with a « major release » approximately every 6 months.
 - Release 9.0.0 : October 27, 2004
 - Release 10.0.0 : 16 Feb 2005
 - Bug Fix releases (9.0.1, 9.0.2...) + incremental builds for test-beam
- ❖ Every night the release in construction is built (« nightlies ») and content kept for a week (useful only for developers of code in the release)



Packages

a way of grouping related code

- ❖ Typically, one package ↔ one library which is dynamically loaded at run time. (there is only one very small athena executable for all applications)
- ❖ For Example, CSC_Digitization contains code for:
 - Taking a MC Hit and producing a Digit (4 ADC samples).
 - The package has a structure:
 - ✧ CSC_Digitization/src : contains *.cxx files
 - ✧ CSC_Digitization/CscDigitization : *.h files
 - ✧ CSC_Digitization/share : jobOption files (no code)
 - ✧ CSC_Digitization/cmt : requirement file
- ❖ Packages may depend on other packages:
 - ✓ CSC_Digitization depends on MuonIdHelpers, MuonDigitContainer, etc.
 - ✓ Dependencies specified in requirement file
 - ✓ “depends” means in most cases “uses object defined in other packages” ↔ “uses header file in other packages”
 - ✓ “depends” **does not mean**: need other packages to be run beforehand
- ❖ Dependency is uni-directional.
 - Packages at the bottom of the chain must be very robust.



Brief tour of the web/ documentation

- ❖ Mailing lists (sw-help, sw-reconstruction, sw-developers, atlas-phys-analysis-tools)
- ❖ Reconstruction web page
- ❖ Following tools from the software developer web page:
 - Howto's ! A must-read
 - viewcvs/lxr ! Browsing the code in cvs
 - Doxygen ! Simple code documentation
 - Savannah ! Bug reporting system
 - release status and plans



Athena is a Framework:

- ❖ The software that makes sure your code
 - Runs at the right time
 - With the right input data
 - And takes care of any output



Athena Terminology [1]

❖ Algorithm:

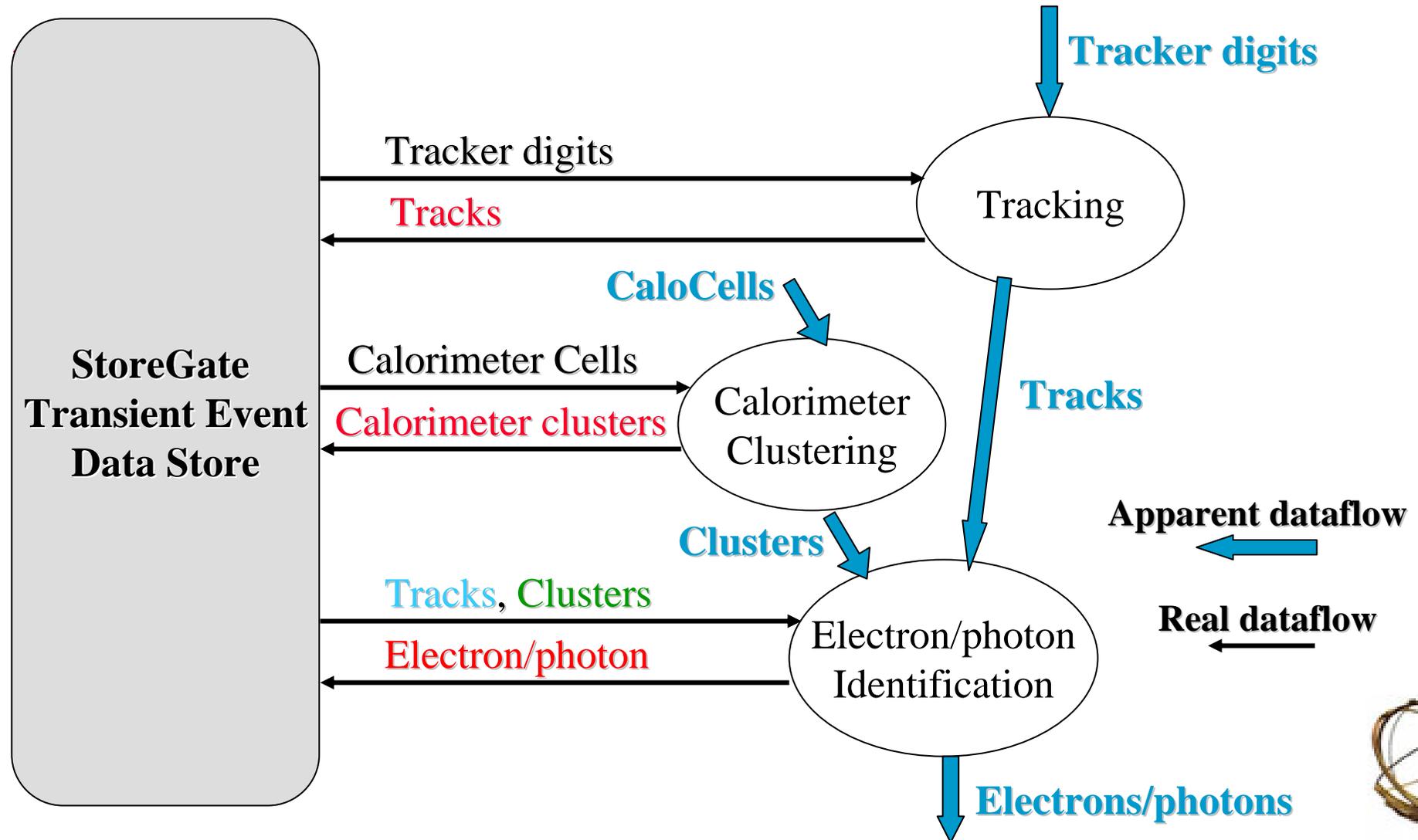
- User application building block, visible & controlled by framework.
 - May delegate processing to AlgTools
 - inherits from Algorithm class
 - Implements three methods for invocation by framework :
 - ✓ initialize(), execute(), finalize()
- For example, the analysis skeleton algorithm

❖ Data Object:

- The result of your algorithm that is posted publicly and can serve as an input to a subsequent algorithm.
 - ✓ e.g., Collection containing Cluster Objects
- Data Objects managed by a Transient Store : aka StoreGate
- Many different type of stores:
 - ✓ Event Store, Detector Store



Algorithm & Data Flow



ESD, AOD, ... streams

- ❖ At the end of each event, the reconstruction output is written into several streams:
 - ESD = Event Summary Data
 - Contains intermediate reconstruction output such as
 - ✓ Tracks, Calorimeter cells, Calorimeter clusters
 - ✓ egamma, jets, muons, Missing ET, ...
 - AOD = Analysis Object Data
 - Container particle level information
 - ✓ electron, photons, b-jets, muons, MissingET, ...
 - ✓ Which allows you to do basic analysis with the ability to navigate back to parent objects if needed
 - Ntuples?
 - ✓ Should be used as a validation tool for ESD and AOD analyses



Athena Terminology [2]

❖ Services

- Globally available software components providing specific framework capabilities, e.g., Message service, Histogram service, etc

❖ Data Converters

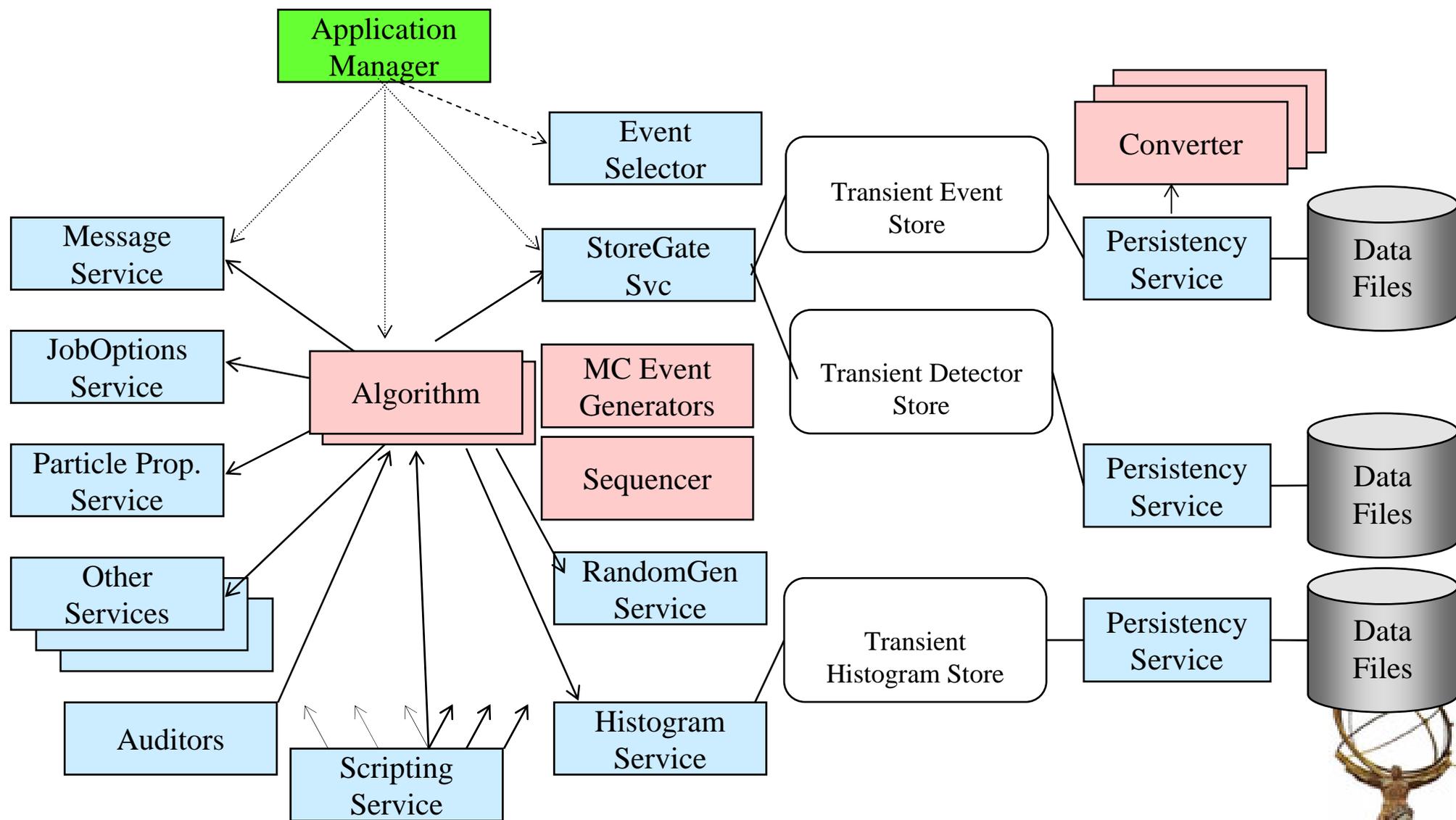
- Provides explicit/implicit conversion from/to persistent data format to/from transient data
- Decouple Algorithm code from underlying persistency mechanism(s)

❖ Properties

- Control and data parameters for Algorithms and Services. Allow for run-time configuration. Specified via startup text files (jobOptions), Python scripts or from the scripting language shell



Athena-Gaudi Object Diagram



PC4

updated the diagram to reflect the way we access the stores. Also added (new) RandomGenSvc (AtRndmGenSvc.h)

Paolo Calaiura, 11/21/2002

Athena Terminology [3]

❖ Job Options files

- Conventional python scripts (default jobOptions.py) used to control an Athena application configuration at run-time

❖ Auditors

- Monitor various aspects of other framework components
 - ✓ NameAuditor, ChronoAuditor, MemoryAuditor, MemStatAuditor, etc

❖ Sequences

- Lists of *members* Algorithms managed by a Sequencer.
- Sequences can be nested. Default behavior: the Sequencer terminates a sequence when an event fails a filter. The *StopOverride* property overrides the default behavior.

❖ Filters

- Event selection criteria.



Accessing Services

- ❖ Within the Algorithm, services are readily accessible.
- ❖ Access to some of the “basic” services are pre-defined PC5
 - `msgSvc()`
 - `histoSvc()`
 - `ntupleSvc()`
- ❖ Access to most other services must be located:
`StatusCode sc = service(“StoreGateSvc”, m_eventStore);`
 - Retrieves the pointer to StoreGate Service
 - Then you use `m_eventStore` in your algorithm to access data objects
 - Typically done in `initialize()` of your algorithm and cached

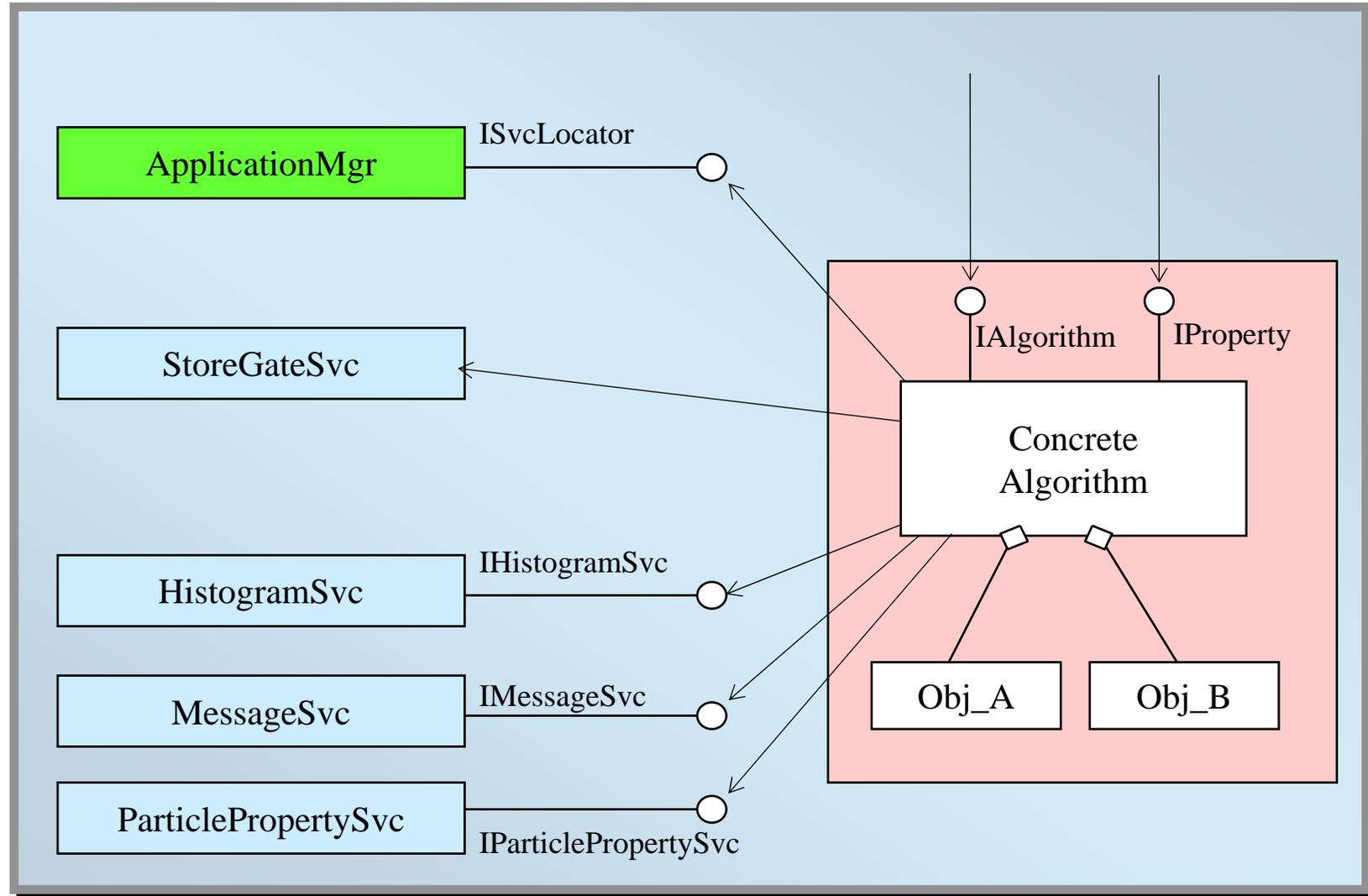


PC5

evtSvc and detSvc are LHCb-only

Paolo Calaiura, 11/21/2002

Accessing Services



PC6

Updated to reflect SG usage
Paolo Calaiura, 11/21/2002

JobOptions

- ❖ The jobOptions file specifies the run-time configuration of your algorithm
 - Specifies all the services needed and their configuration
 - Determines what algorithms need to be run
 - In what order
 - Specifies the properties of the algorithm
- Control over :
 - ✓ Message Output Level
 - ✓ Number of Events to process
 - ✓ Input file names.
 - ✓ Output file names, objects to save etc.
 - ✓ And so on.



Some Common Entries in jobOptions.py

❖ Including other python scripts:

```
include ( "RecExCommon/RecExCommon_flags.py" )  
include( "CaloRec/CaloCluster_jobOptions.py" )
```

❖ Component libraries to be loaded

theApp.DLLs += [<comma separated array of string>
e.g. theApp.DLLs += ["MuonByteStream", "MuonAlgs"]

❖ Top level algorithms: "Type/ObjectName"

theApp.TopAlg += [<comma separated Array of string>
e.g. theApp.TopAlg += ["MuonBuilderAlg"]

❖ Maximum number of events to execute

```
theApp.EvtMax      =      100  
RunNumber         =      200100
```

→ internally, theApp.RunNumber = RunNumber (global flag)

❖ Comments

Preceded by #



Configuring an Algorithm

- ❖ If you have an Athena algorithm MyAlg, you need to specify the following in the jobOptions:
 - ❖ `theApp.DLLs += ["MyAlgLib"]`
 - Name of the library where MyAlg is located
 - This is typically the package name
 - ❖ `theApp.TopAlg += ["MyAlg/MyAlgName"]`
 - MyAlg is your Algorithm class name
 - MyAlgName is any name you give it
 - Algorithms are run in the sequence they appear in the jobOptions
 - ❖ Specify property of the algorithm
 - `MyAlgName = Algorithm("MyAlgName")`
 - `MyAlgName.someProperty = property_value`



Running athena

- ❖ **athena**
 - By default, it looks for jobOptions.py
- ❖ **athena MyJobOptions.py**
 - Athena will use MyJobOptions.py for input configuration
- ❖ **athena MyJobOptions.py >& athena.log**
 - Directs all screen printout to athena.log
- ❖ **athena -s MyJobOptions.py >& athena.log**
 - -s prints out all jobOptions scripts that are included within
- ❖ **athena -c "EvtMax=10; doWriteESD=True" >& athena.log**
 - Uses the options to overwrite default ones in jobOptions.py
 - Only works today with RecExCommon
- ❖ **athena -i MyJobOptions.py**
 - Interactive mode of running athena



Standard Top Level JobOptions

- ❖ **RecExCommon_topOptions.py**
 - Use to run full ATLAS Reconstruction on G3 or G4 simulation
 - Outputs ESD and AOD data and minimal ntuples
- ❖ **RecExTB_Combined_2004_jobOptions.py**
 - Use to run reconstruction software on TB data and TB simulation
 - Outputs ESD data
- ❖ **TBAnalysis_topOptions.py**
 - Used to analyze the output ESD data from RecExTB
 - Clients can plug in their own analysis algorithms
 - Outputs user ntuples
- ❖ **Other top level jobOptions available for :**
 - ATLAS and TB : Simulation & Digitization
 - That output POOL files with relevant data



Today

❖ You will learn how to :

- Setup the software environment and compile your code
- Familiarize yourself how to configure your code with jobOptions
- use RecExCommon and UserAnalysis to reconstruct DC2 simulated data:
 - ✓ Produce ESD and AOD
 - ✓ And look at ntuples
- Write an analysis algorithm in Athena on AOD data as input
 - ✓ And produce your own analysis ntuples
- Write simple python based analysis scripts with AOD data as input
 - ✓ Using Athena and ROOT features interactively

