The ATLAS Computing Workbook

02 May 2009

Contributors:
Christian Arnault, Ketevi Assamagan, Marcello B, Se Boeser, James Catmore, Chris Collins Tooth, Markus Cristinziani, Simon Dean, Eric Eisenhandler, Simon George, Ricardo Goncalo, Grant Gorfine, Andrew Hamilton, Traudl Hansl-Kozanecka, Vivek Jain, Roger Jones, Traudl Kozanecki, Gernot Krobath, Mario Lassnig, Wim Lavrijsen, Steve Lloyd, Edward Moyse, David Quarrie, David Rousseau, Oxana Smirnova, Juergen Thomas, Nathan Triplett, Ikuo Ueda, Pete Watkins, Monika Wielers, Saul Youssef.
TWiki to PDF Converter TWiki2pdf Version 2.3.7 by Steve Lloyd.

Contents

Preface

Acknowledgements

The ATLAS Workbook is loosely modelled on the BaBar Workbook and the OPAL Primer. The Workbook is the work of many people who have contributed knowingly or unknowingly. Some of the early material is based directly or indirectly on the tutorials of Wim Lavrijsen (http://wlav.home.cern.ch/wlav/tutorials/), James Catmore and Chris Collins-Tooth (http://agenda.cern.ch/fullAgenda.php?ida=a05403) and Ketevi Assamagan (http://www.usatlas.bnl.gov/PAT/tutorial904.html). Other acknowledgements appear at the foot of each page.

Purpose of the Workbook

The purpose of the Workbook is to introduce new members of the ATLAS Collaboration to the ATLAS offline software. The main emphasis is on the requirements of physicists doing analysis rather than those developing the code, however it should serve as a useful introduction to new developers.

Once you have gone through this workbook, you should look at the AtlasProtected.PhysicsAnalysisWorkBook, which covers in much more detail the analysis tools and procedures used by ATLAS.

Using the Workbook

The primary source of the Workbook is a number of ATLAS Twiki pages starting with WorkBook. Each page or section has a **Responsible** person who should be the main contact for queries and suggested modifications. Each page is also reviewed periodically to see if the content is still valid. This is especially important if the content hasn't changed for a long time but may still be correct. There is also a progress indicator which gives the user an idea how complete the topic is in terms of the amount of content suitable for the Workbook (rather than everything that could be written on the topic). The scale goes from empty - 0 (blank) through 1 and 2 (red), 3 and 4 (yellow) to complete - 5 (green). On the top WorkBook page there is a link to a page that summarises each other page in terms of modification and review dates and content.

Each page has icons that allow the Workbook to be converted into PDF for higher quality printing:

- converts the whole Workbook to PDF (may take some time).
- converts one chapter to PDF .
- converts the current page only.

The conversion to PDF is done using the Add-On package TWiki2Pdf .

If you have problems with any of the recipes in the Workbook please try the General Offline Help HyperNews Forum. If you have suggestions or would like to contribute to the development of the Workbook please try the Documentation and Communication Forum.

Workbook Formatting Rules

The primary source of the Workbook pages are stored as TWiki topics. In order to produce a coherent package, a few simple rules need to be adhered to:

- All Workbook topic names should be of the form **WorkBookSomeTopicName** i.e. an obvious name preceded by **WorkBook**.
- Topics beginning with **WorkBookArchive** are sections referring to old releases for instance and won't appear in the pdf version except as links.
- There should then follow a first level header with this name such as **Some Topic Name**.
- Two variables should be included to link to the PDF versions - **%PDFALL%** and **%PDFTHIS%**.
- The variable **%COMPLETEn%** where **n** is 0 to 5 is used to indicate how complete the topic is.
- At the bottom, signatures should be inserted each time there is a significant change.
- The responsible person should be indicated by **%RESPONSIBLE% Main.WhoEver**.
- The last time the topic was reviewed should be indicated by **%REVIEW% Main.WhoEver - Date**
- Wherever possible TWiki editing should be used rather than raw HTML.

An example of these would be:

```
---+ Some Topic Name
%PDFALL% %PDFTHIS%
%COMPLETE2%

Some introductory text

---++ Subsection Heading

More text

-- Main.SignatureName - 25 Feb 2005

%RESPONSIBLE% Main.WhoEver1 %BR%
%REVIEW% Main.WhoEver2 - 25 Feb 2005
```

Indexing

You can make entries in the index using the following at the beginning of a line:

```
#SomeWord<index entry="index entry"/>
```

where `SomeWord` is any WikiWord that acts as an anchor and `index entry` is the text you want in the index. If this is of the form `entry - sub entry`, then `sub entry` becomes a sub item of `entry` in the PDF version.

Figures

You can include higher resolution figures in the PDF version by including a `pdfsrc` on the `img` tag e.g.

```
<img src="%ATTACHURLPATH%/NormalFigure.jpg" pdfsrc="%ATTACHURLPATH%/
    HighResFigure.eps" />
```

You can control the width independently of the normal figure by using `pdfwidth` instead/as well as `width`.

Including Text

You can include pieces of other TWiki pages by including the following in the file to be included (as many times as necessary):

```
<!--STARTWORKBOOK-->
Text to be included
...
<!--STOPWORKBOOK-->
```

and the following in the Workbook page where you want it included where the number in brackets refers to which occurance of the `STARTWORKBOOK/STOPWORKBOOK` you want to include and the last bit picks up the signatures from the end:

```
%INCLUDE{"SomeTWikiTopic" pattern="(?:.*?<!--STOPWORKBOOK-->){0}.*?<!--
    STARTWORKBOOK-->(.*?)<!--STOPWORKBOOK-->.*"}%
%INCLUDE{"SomeTWikiTopic" pattern="(?:.*?<!--STOPWORKBOOK-->){1}.*?<!--
    STARTWORKBOOK-->(.*?)<!--STOPWORKBOOK-->.*"}%
...
%INCLUDE{"SomeTWikiTopic" pattern=".*?(\-\- Main\..*)"}%
```

Variables

To avoid extensive editing when things like release numbers change, the following variables are defined:

- %WBRELEASE% - The latest production release - currently set to 14.2.23
- %WBHELLOWORLD% - The latest production version of AthExHelloWorld - currently set to 01-01-01
- %WBUSERANALYSIS% - The latest production release of UserAnalysis - currently set to 00-13-06
- %WBCMTVERSION% - The latest production version of CMT - currently set to v1r20p20080222
- %WBJIVEXML% - The latest production version of JiveXML - currently set to 00-06-49
- %WBPACMAN% - The latest production version of Pacman - currently set to 3.25
- %WBDETECTORDESC% - The latest detector description - currently set to ATLAS-CSC-02-02-00
- %WBDBRELEASE% - The latest database release - currently set to 3.1.1

Since TWiki variables are not expanded inside `<verbatim>...</verbatim>` you should use `<pre>...</pre>` instead.

Chapter 1

Introduction

1.1 The ATLAS Experiment

1.1.1 Introduction

ATLAS (A Toroidal LHC ApparatuS) is one of two General Purpose Detectors at the CERN Large Hadron Collider (LHC). The LHC will collide 7 TeV protons together with a centre of mass energy of 14 TeV and a design luminosity of $10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$. The bunch crossing time will be 25ns and at full luminosity there will be approximately 22 proton-proton collisions per bunch crossing. The ATLAS Detector is situated at Point 1, directly opposite the CERN main entrance. The ATLAS detector consists of four major components, the Inner Tracker which measures the momentum of each charged particle, the Calorimeter which measures the energies carried by the particles, the Muon spectrometer which identifies and measures muons and the Magnet system that bends charged particles for momentum measurement. The detector is a cylinder with a total length of 42 m and a radius of 11 m and weighs approximately 7000 tonnes.

1.1.2 The ATLAS Coordinate System

The ATLAS Coordinate System is a right-handed system with the x-axis pointing to the centre of the LHC ring, the z-axis following the beam direction and the y-axis going upwards. In Point 1, positive z points towards Point 8 with a slope of -1.23%. The azimuthal angle $\phi = 0$ corresponds to the positive x-axis and ϕ increases clock-wise looking into the positive z direction. ϕ is measured in the range [-π, $+\pi$]. The polar angle θ is measured from the positive z axis. Pseudorapidity, η, is defined by

$$\eta = -\log\left(\tan\frac{\theta}{2}\right)$$

Transverse momentum, p_T, is defined as the momentum perpendicular to the LHC beam axis.

Lengths are measured in mm and energies in MeV. For more information see AtlasUnits.

1.1.3 Inner Detector

Pixel Detector

A Pixel sensor is a 16.4 x 60.8 mm wafer of silicon with 46,080 pixels, 50 x 400 microns each. The barrel part of the pixel detector consists of the 3 cylindrical layers with the radial positions of 50.5 mm, 88.5 mm and 122.5 mm respectively. These three barrel layers are made of identical staves inclined with azimuthal angle of 20 degrees. There are 22, 38 and 52 staves in each of these layers respectively. Each stave is composed of 13 pixel modules. There are three disks on each side of the forward regions. One disk is made of 8 sectors, with 6 modules in each sector. Disk modules are identical to the barrel modules, except the connecting cables.

Semiconductor Tracker (SCT)

The SCT system is designed to provide eight precision measurements per track in the intermediate radial range, contributing to the measurement of momentum, impact parameter and vertex position. In the barrel SCT eight layers of silicon microstrip detectors provide precision points in the r-phi and z coordinates, using small angle stereo to obtain the z-measurement. Each silicon detector is 6.36 x 6.40 cm with 768 readout strips of 80 micron pitch. The barrel modules are mounted on carbon-fibre cylinders at radii of 30.0, 37.3, 44.7, and 52.0 cm. The end-cap modules are very similar in construction but use tapered strips with one set aligned radially. The SCT covers $\mid \eta \mid < 2.5$.

Transition Radiation Tracker (TRT)

The outer tracker of ATLAS is a combined straw tracker and transition radiation detector. The barrel part contains 52544 axial straws of about 150 cm length at radii between 56 cm and 107 cm. The end-caps contain a total of 319488 radial straws at radii between 64 cm and 103 cm (inner end-caps), respectively 48 cm and 103 cm (outer end-caps).

The TRT provides on average 36 two-dimensional measurement points with 0.170 mm resolution for charged particle tracks with $\mid \eta \mid < 2.5$ and $p_T > 0.5$ GeV.

1.1.4 Calorimeter

Liquid Argon Calorimeter

The Liquid Argon (LAr) Calorimeter is divided into several components: an electromagnetic sampling calorimeter with 'accordion-shaped' lead electrodes in the barrel and in the endcaps, a hadronic calorimeter using flat copper electrodes in the endcaps, and a forward calorimeter close to the beampipe in the endcap made from copper and tungsten. In addition, presamplers consisting of one layer of LAr in front of the electromagnetic calorimeter help to correct for the energy loss in front of the calorimeter (mainly due to cryostat walls and the barrel solenoid).

Tile Calorimeter

The Tile Calorimeter is a large hadronic sampling calorimeter which makes use of steel as the absorber material and scintillating plates read out by wavelength shifting (WLS) fibres as the active medium. It covers the central range $\mid \eta \mid < 1.7$. The new feature of its design is the orientation of the scintillating tiles which are placed in planes perpendicular to the colliding beams and are staggered in depth. A good sampling homogeneity is obtained when the calorimeter is placed behind an electromagnetic compartment and a coil equivalent to a total of about two interaction lengths of material. The Tile Calorimeter consists of a cylindrical structure with an inner radius of 2280 mm and an outer radius of 4230 mm. It is subdivided into a 5640 mm long central barrel and two 2910 mm extended barrels as shown in the Figure below. The thickness of the calorimeter in the gap is improved, which has the same segmentation as the rest of the calorimeter. The total number of channels is about 10000.

1.1.5 Muon System

In the barrel region ($| \eta |< 1.0$), which is covered by the large barrel toroid system, muons are measured in three layers of chambers around the beam axis using precision Monitored Drift Tubes (MDTs) and fast Resistive Plate Chambers (RPCs). In regions of larger pseudorapidity, also three layers of chambers are installed, but vertically. Here Thin Gap Chambers (TGCs) are used for triggering. The precision measurement of muons is again done with MDTs, except for the innermost ring of the inner station of the end caps and for $| \eta |> 2$, where high particle fluxes require the more radiation tolerant Cathode Strip Chamber (CSC) technology.

In the barrel of the ATLAS muon system, the muon chambers are installed in three cylinders concentric with the beam axis at radii of about 5, 7.5 and 10 m. They are arranged to form projective towers pointing to the nominal interaction vertex. In the end caps, the distance in z from the vertex is about 7, 10 and 14 m for the three layers.

1.1.6 Magnet System

Central Solenoid

The central ATLAS solenoid has a length of 5.3 m with a bore of 2.4 m. The conductor is a composite that consists of a flat superconducting cable located in the center of an aluminum stabaliser with rectangular cross-section. It is designed to provide a field of 2 T in the central tracking volume with a peak magnetic field of 2.6 T. To reduce the material build-up the solenoid shares the cryostat with the liquid argon calorimeter.

Toroid Magnet

The ATLAS Toroid Magnet system consists of eight Barrel coils housed in separate cryostats and two End-Cap cryostats housing eight coils each. The End-Cap coils systems are rotated by 22.5 with respect to the Barrel Toroids in order to provide radial overlap and to optimise the bending power in the interface regions of both coil systems.

Complete: 🟩
Responsible: Steve Lloyd
Author: Steve Lloyd 02 Apr 2005
Contributors: Steve Lloyd
Last significantly modified by: Steve Lloyd 02 Apr 2005
Not yet reviewed

1.2 The ATLAS Computing Model

The ATLAS Collaboration has developed a set of software and middleware tools that enable access to data for physics analysis purposes to all members of the collaboration, independently of their geographical location. Main building blocks of this infrastructure are:

The Athena software framework, with its associated modular structure of the event data model, including the software for:

- Event simulation
- Event trigger

- Event reconstruction
- Physics analysis tools

The Distributed Computing tools are built on top of Grid middleware:

- The Distributed Data Management (DDM) system
- The Distributed Production System
- The Ganga/pAthena frameworks for distributed analysis on the Grid *Monitoring and accounting

DDM is the central link between all components as data access is needed for any processing and analysis step!

1.2.1 The Event Data Model (EDM)

The Event Data Model defines a number of different data formats:

- **RAW**:
  - "ByteStream&#8221; format, ∼1.6 MB/event
- **ESD** (Event Summary Data):
  - Full output of reconstruction in object (POOL/ROOT) format:
    - Tracks (and their hits), Calo Clusters, Calo Cells, combined reconstruction objects etc.
  - Nominal size 1 MB/event initially, to decrease as the understanding of the detector improves
    - Compromise &#8220;being able to do everything on the ESD&#8221; and &#8220;not storage for oversize events&#8221;
- **AOD** (Analysis Object Data):
  - Summary of event reconstruction with &#8220;physics&#8221; (POOL/ROOT) objects:
    - electrons, muons, jets, etc.
  - Nominal size 100 kB/event (currently roughly double that)
- **DPD** (Derived Physics Data):
  - Skimmed/slimmed/thinned events + other useful &#8220;user&#8221; data derived from AODs and conditions data
    - DPerfD is mainly skimmed ESD
  - Nominally 10 kB/event on average
    - Large variations depending on physics channels
- **TAG**:
  - Database (or ROOT files) used to quickly select events in AOD and/or ESD files

1.2.2 The Operational Model

- **Tier-0** (CERN):
  - Copy RAW data to CERN Castor for archival & Tier-1s for storage and reprocessing
  - Run first-pass calibration/alignment
  - Run first-pass reconstruction (within 48 hrs)
  - Distribute reconstruction output (ESDs, AODs, DPDs & TAGS) to Tier-1s
- **Tier-1** (x10):
  - Store and take care of a fraction of RAW data (forever)
  - Run &#8220;slow&#8221; calibration/alignment procedures
  - Rerun reconstruction with better calib/align and/or algorithms
  - Distribute reconstruction output to Tier-2s
  - Keep current versions of ESDs and AODs on disk for analysis
  - Run large-scale event selection and analysis jobs for physics and detector groups

  - Looks like some user access will be granted, but limited and NO ACCESS TO TAPE or LONG TERM STORAGE
- **Tier-2** (x∼35):
  - Run analysis jobs (mainly AOD and DPD)
  - Run simulation (and calibration/alignment when/where appropriate)
  - Keep current versions of AODs and samples of other data types on disk for analysis
- **Tier-3**:
  - Provide access to Grid resources and local storage for end-user data
  - Contribute CPU cycles for simulation and analysis if/when possible

---

Originally taken from various talks by Roger Jones e.g. `http://indico.cern.ch/getFile.py/access?contribId=1&resId=1&materialId=slides&confId=48759`.

Complete: 🟨🟨🟨⬜⬜
Responsible: Steve Lloyd
Author: Roger Jones 16 Jan 2008
Contributors: Roger Jones, Steve Lloyd
Last significantly modified by: Steve Lloyd 16 Jan 2008
Not yet reviewed

1.3 The Athena Framework

Athena is a **control framework** and is a concrete implementation of an underlying **architecture** called **Gaudi**. The Gaudi project was originally developed by LHCb. Nowadays the Gaudi project is a kernel of software common to both experiments and co-developed, while Athena is the sum of this kernel plus ATLAS-specific enhancements. Guides are available from AthenaFramework.

1.3.1 What is a Framework?

- A skeleton of an application into which developers plug in their code
- Provides most of the common functionality and communications between different components
- Embodies the underlying design and philosophy of the software
- Encourages a common approach
- Factors out common functionality for re-use

1.3.2 Athena Terminology

Algorithm:

- User application building block, visible & controlled by framework.
- May delegate processing to AlgTools
- Inherits from Algorithm class
- Implements methods for invocation by framework : initialize(), execute(), beginRun(), endRun() and finalize()

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 track objects
- List of particles that passed some selection
- Data Objects managed by a Transient Store : aka StoreGate
- Several different type of stores: Event Store, Detector Store

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

Data Converters:
- Provides explicit/implicit conversion from/to persistent data format (in files) to/from transient data (in memory)
- Decouple Algorithm code from underlying persistency mechanism(s)

Properties:
- Control and data parameters for Algorithms and Services. Allow for run-time configuration. Set in the Python scripts used to run Athena aka JobOptions

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

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.

Originally taken from http://agenda.cern.ch/fullAgenda.php?ida=a05403

Complete: 🟨🟨🟨⬜⬜
Responsible: Steve Lloyd
Author: Simon George 03 Feb 2005
Contributors: Simon George, Wim Lavrijsen, Steve Lloyd
Last significantly modified by: Steve Lloyd 23 Jan 2006
Last reviewed by: Wim Lavrijsen 23 Jan 2006

1.4 Athena Job Options

The Job Options file specifies the run-time configuration of your algorithm. It specifies all the services needed and their configuration. It determines what algorithms need to be run, in what order and specifies the properties of the algorithms. For further details see http://cern.ch/wlav/joboptions.

The Job Options give Control over :
- Message Output Level
- Number of Events to process
- Input file names.

- Output file names, objects to save etc.
- And so on.

The default job options files is jobOptions.py

1.4.1 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>]
theApp.DLLs += [" CaloRec", "LArClusterRec","TileRecAlgs" ]
```

Top level algorithms: `["Type/ObjectName"]`

```
theApp.TopAlg += [<comma separated Array of string>]
theApp.TopAlg += [" CaloClusterMaker/CaloSWClusterMaker" ]
```

Maximum number of events to execute

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

internally `theApp.RunNumber = RunNumber` (global flag)

Comments are preceded by #.

1.4.2 Configuring an Algorithm

If you have an Athena algorithm `MyAlg`, you need to specify the following in the jobOptions:

```
theApp.DLLs += [" MyAlgLib"]
```

with 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 for use below.

Note that you must use += in these and not = as these are appended to the lists of existing algorithms.

Algorithms are run in the sequence they appear in the jobOptions

To specify some property of the algorithm:

```
MyAlgName = Algorithm("MyAlgName")
MyAlgName.someProperty = property_value
```

1.4.3 Global Flags

There are many flags that are set to a default value in the jobOptions and can be over-written to suit your needs. These major flags are used to control the execution process:

```
theApp.EvtMax = 100    # maximum number of events
RunNumber = 201001    # run number
doWriteESD = False    # do not write ESD output
```

Flags to control which sub-system reconstruction to run:

```
doInDet = False
doLar = True
doTile = True
```

plus Sub-System specific flags.

Originally taken from the ATLAS-UK e-Science Training Course Introductory Talk by Simon George

Complete: ▨▨▨□
Responsible: Steve Lloyd
Author: Simon George 03 Feb 2005
Contributors: Simon George, Wim Lavrijsen, Steve Lloyd
Last significantly modified by: Steve Lloyd 23 Jan 2006
Last reviewed by: Wim Lavrijsen 23 Jan 2006

1.5 Projects, Packages and Releases

ATLAS Software is organised into a hierarchical structure of **Projects** and **Packages**. Each package has a Tag number that distingushes different versions. Each version of a project has a release number and there is an overall Release number that identifies a complete collection of packages. Package Tags may be updated frequently but new production releases only occur every few months. There may be more frequent development releases in between. There are Nightly Builds of the most recent packages for developers. The concept of projects was introduced in ATLAS Release 12. Before that all the packages were effectively in one "project". More information can be found in AthenaReleases.

1.5.1 Projects

The ATLAS offline has been split into the following set of projects (in alphabetical order):

- **AtlasAnalysis**. This contains Tools, Algorithms and Services associated with physics analysis, monitoring and the event display. It depends upon the AtlasTrigger project.
- **AtlasConditions**. This project contains detector description, geometry and calibration information. It depends upon the AtlasCore project.
- **AtlasCore**. This contains core components and services (e.g. Athena and StoreGate). It depends upon the Gaudi framework project, and the LCGCMTT project which provides links to LCGG supported external software packages such as POOL and ROOT.
- **AtlasEvent**. This project contains the event data model (EDM) and support classes. It depends upon the AtlasConditions project.

- **AtlasOffline**. This is the project that provides the default entrypoint into the project hierarchy for interactive use. It depends upon the AtlasAnalysis and AtlasSimulation projects via explicit dependencies. By default it contains essentially no packages, but is a placeholder whereby bugfix releases may be created. The release number of this project **is** the ATLAS offline release number.
- **AtlasProduction**. This is a top-level project for production use of the ATLAS offline release. It depends upon all other projects . Only a few packages are assigned to this project, in particular those that allow global testing of the complete release in a production enironment (e.g. KitValidation). By convention the release number of this project is the same as the ATLAS offline release number, although a forth digit (i.j.k.l) denotes a patch that is applied for production use only. See also AtlasProtected.WorkingWithPatchedReleases
- **AtlasReconstruction**. This contains Tools, Algorithms and Services used for reconstruction and the fast simulation package Atlfast. It depends upon the AtlasEvent project.
- **AtlasSimulation**. This contains the Geant4 simulation, the generators, pile-up and digitization Tools, Algorithms and Services. It depends upon the AtlasEvent project.
- **AtlasTrigger**. This contains Tools, Algorithms and Services associated with the high level trigger. It depends upon the AtlasReconstruction project.

The project dependencies are shown in the following diagram:

```
AtlasProduction
                     |
                AtlasOffline
                     |
         +-----------+----------+
         |                      |
    AtlasAnalysis               |
         |                      |
     AtlasTrigger               |
         |                      |
  AtlasReconstruction      AtlasSimulation
         |                      |
         +-----------+----------+
                     |
                 AtlasEvent
                     |
              AtlasConditions
                     |
                 AtlasCore
                     |
                   Gaudi
                     |
                   LCGCMT
```

where each of the projects depends upon those below it.

For more information about Projects see WorkingWithProjectBuilds.

1.5.2 Packages

Inside these projects, the software is organised in a number of packages. The main domains are:

- Generators, Simulation, Trigger, Reconstruction, PhysicsAnalysis
- InnerDetectorSoftware, MuonSoftware, LArCalorimeter, TileCalorimeter

- CommonTrackingSoftware, Calorimeter
- Control, DataBase, DetectorDescription, Monitoring, MagneticField
- AtlasSettings, AtlasRelease, AtlasPolicy, AtlasCxxPolicy, AtlasTest, Tools, Utilities
- External

These are subdivided, e.g. InnerDetectorSoftware has about 20 sub-packages.

Most packages shown in the hierachy are "container" packages used for structuring the software and managing versions in releases At the bottom of the tree are "leaf" packages

- Contain software
- Contain meta-information for the configuration management tool (CMT)
- Have a common layout of subdirectories for headers, source, config, data

More about package structure: `http://atlas-computing.web.cern.ch/atlas-computing/documentation/swDoc/swArchive/PackageStructure.txt`

Package Layout

Each package is organized into several sub-directories (not all are necessary):

- `cmt` : contains requirements + setup files. This is where you build your code.
- `<PackageName>` : contains the C++ header files.
- `src` : contains the C++ source files.
- `share` : contains job option (.py) files.
- `run` : where you run your jobs.
- `python` : any additional python source.

For instance the **UserAnalysis** package contains (among other things) the following:

```
PhysicsAnalysis/AnalysisCommon/UserAnalysis/
cmt/requirements
UserAnalysis/AnalysisSkeleton.h
src/AnalysisSkeleton.cxx
components/UserAnalysis_entries.cxx
components/UserAnalysis_load.cxx
share/AnalysisSkeleton_jobOptions.py
doc/
run/
i686-slc3-gcc323-opt/
```

`AnalysisSkeleton.cxx` is the C++ source file and `AnalysisSkeleton.h` the header file.
`AnalysisSkeleton_jobOptions.py` are suitable Job Options.
`UserAnalysis_entries.cxx` and `UserAnalysis_load.cxx` are used to create entry points into a dynamically loaded library, so you can say at runtime "load this algorithm" and it will be found in the correct library and the lib will be loaded into memory at that time.

Note: Prior to Release 12 there used to be another level in the directory structure i.e.
`PhysicsAnalysis/AnalysisCommon/UserAnalysis/UserAnalysis-NN-NN-NN`

Tags

If you check out a package (using `cmt co`) you get the latest version of that package which may not necessarily be compatible with older releases in which case it is better to explicitly choose which version of the package you want (using `cmt co -r PackageName-XX-YY-ZZ`). The number `XX-YY-ZZ` is called the **Tag** for that version of the package.

To find out the relevant tag for a package in a given release, go to http://atlas-computing.web.cern.ch/atlas-computing/links/buildDirectory and drill down through the correct release to the package you want and then see what tag it has.

If you have already set up AtlasOffline project as is described in WorkBookSetAccount, you can use `cmt show versions` command to get the proper version of the package. For example, to find out the relevant tag for `PhysicsAnalysis/AnalysisCommon/UserAnalysis`

```
cmt show versions PhysicsAnalysis/AnalysisCommon/UserAnalysis
```

It will display

```
PhysicsAnalysis/AnalysisCommon/UserAnalysis UserAnalysis-00-08-19 /atlas/
    AtlasAnalysis/2.0.3
```

for Athena 12.0.3 release.

`cmt show versions` command requires the full package hierarchy. What if you know only the package name instead of the full path name, for example, `UserAnalysis`? If the version of `cmt` is v1r18p20060606 or above, you can get the same information by using `cmt show packages` with `grep` command:

```
cmt show packages | grep 'UserAnalysis '
```

It will show

```
UserAnalysis UserAnalysis-00-08-19 /atlas/AtlasAnalysis/2.0.3/PhysicsAnalysis/
    AnalysisCommon
```

From the above output, we know that `UserAnalysis` is a leaf package of `PhysicsAnalysis/AnalysisCommon` and its version for this Athena release is `UserAnalysis-00-08-19`.

Release Numbers

The ATLAS software framework is continually updated to allow developers to insert new code and amend bugs. Periodically the whole software is 'released' and given a "Release Number". This is a group of three integers separated by dots, where the first number is the major release, the second indicates branchings, and the third indicates minor changes. You will frequently find yourself updating to a new release. Up to Release 13 the following was true:

- The N.0.n series were "Production" Releases.
- The N.m.0 series were Developer Releases.
- The N.m.n series were special releases such as for the Test Beam.

From Release 14 onwards the distinction between production and development has been removed.

Occasionally a release needs to be *patched* after release to correct some bugs. For instance release 14.2.23.4 is a patched version of 14.2.23.

Up to Release 12, the Production releases were numbered 12.0.1, 12.0.2 etc. For Release 13 onwards they were numbered 13.0.10, 13.0.20 etc. From Release 14 they are numbered 14.0.0, 14.1.0 etc.

The recipes in this Workbook refer to Release `14.2.23`. A summary of all releases can be found on the Release Status Page.

Originally taken from `https://twiki.cern.ch/twiki/bin/view/AtlasProtected/AthenaBasics` and WorkingWithProjectBuilds.

Complete: ▰▰▰▰▱
Responsible: Steve Lloyd
Author: Simon George 03 Feb 2005
Contributors: Simon George, Steve Lloyd, David Quarrie
Last significantly modified by: Steve Lloyd 01 Aug 2006
Last reviewed by: Steve Lloyd 25 Feb 2005

1.6 Getting Help

The best way to get help with ATLAS software is to use the Hypernews Forums in e-Groups that have replaced HyperNews.

For Grid related problems you should contact atlas-user-support@ggus.org or go to `http://ggus.org/`

To report bugs (not ask for help) you should use Savannah: `https://savannah.cern.ch/bugs/?func=additem&group=atlas-bugs`

There is also an ATLAS Collaborative Tools page with other useful information and links.

Finally, there is a high-level Help page on the left-bar of all ATLAS TWiki pages.

1.6.1 HyperNews

HyperNews was a discussion and annoucement forum system, ported from SLAC, that replaced many of the existing ATLAS mailing lists and discussion pages. It has in turn been replaced by Hypernews Forums in e-Groups .

1.6.2 HyperNews Forums in e-Groups

These have replaced the previous Hypernews Forums and are available at `https://espace.cern.ch/atlas-forums/default.aspx`. For further information see MailingLists. There is a FAQ at `https://espace.cern.ch/atlas-forums/Lists/FAQ/AllItems.aspx`.

For general help there is:

- General Offline Help.

Other useful forums are:

- CERN Computing Announcements for announcements of changes, outages and other problems in the computing at CERN.
- Releases and Distribution Kit Problems for requests for assistance with problems running the software or installing the kit.
- Software Developers Announcements for announcements, which are intended for developers of ATLAS offline software, e.g. the announcement of developer releases.
- Preliminary/unconfirmed Bugs, Problems, Frustrations, Fixes
- Offline SW Development Discussions or the Athena and Package developers to discuss design and other architectural issues. No bug reports or requests for help should be posted to this forum.

1.6.3 Mailing Lists

There are many ATLAS mailing lists. See `https://websvc03.cern.ch/listboxservices/simba2/free/atlas/atlas.aspx` for a full list. They have been migrated from SIMBA to e-Groups - see MailingLists. Note that in order to post messages to ATLAS mailing lists you **must be properly registered as an ATLAS member at CERN** as described in WorkBookGetAccount#CernAccount. More information and solutions to some of the problems you might encounter are described in MailingListIssues.

1.6.4 Web Pages

There are two collections of ATLAS computing web pages:

- The **static** web pages at `http://cern.ch/atlas-computing/computing.php` These are centrally maintained and generally contain long lived information. They act as portal pages to many other sets of web pages.
- The **TWiki** at `https://twiki.cern.ch/twiki/bin/view/Atlas/AtlasComputing` These pages are in principle editable by anybody who has a CERN AFS password and anybody can contribute. They contain detailed information on most aspects of ATLAS computing although people need to take care that the information is still valid.

1.6.5 Tutorials

Comprehensive lists of tutorials are available:

- Summary list of tutorials

1.6.6 Phone Conferences

Standard Phone Conferences

To reserve a phone conference send a message to Standard.Telephone@cern.ch. Specify:

- Date
- Time from ... to
- Title
- Organizer
- Number of participants

To connect to a conference, dial +41-22-767 7000 and give the title and organizer.

Protected Phone Conferences

An "AudioConf" service allows protection using a code number. To be able to book meetings, you need to sign up, i.e. fill in the Audioconference Account Request (requires AIS account); you have to provide an "authorisation password". There is no charge, but your team account will be used in case of call-back usage. If you do not want your group to pay, you can turn this option off before booking. More information on the system see Start Guide: Automated Audioconferencing Server.

To connect to a conference, dial +41-22-767 6000 and give the access code for the conference you require.

Originally taken from `http://cern.ch/atlas-computing/organization/communication/hypernews.php`

Complete: 🟩
Responsible: Steve Lloyd
Author: Traudl Hansl-Kozanecka 11 Jan 2006
Contributors: Traudl Hansl-Kozanecka, Steve Lloyd
Last significantly modified by: Steve Lloyd 21 Jan 2009
Not yet reviewed

Chapter 2

Getting Started

2.1 Getting an Account

There are a number of methods of running ATLAS software, but the easiest way is to use the CERN Public Login system LXPLUS. The main advantage is that, from this environment, you will have direct access to the AFS file system, on which the ATLAS software **Releases** reside. In addition all the tools you need to develop your software are pre-installed, so you can start work almost immediately. You can also use a pacman **Kit** if it is installed at your institute although you may still need a CERN account in order to checkout packages. If you want to use the Grid you will need a **Grid Certificate**. You only need to use one of these methods to start with e.g. start by using LXPLUS then later use a kit at your home institution and finally use the Grid.

If you are not familiar with using Linux there is a list of **basic Linux Commands** here.

2.1.1 Getting an ATLAS Account at CERN

Contact the ATLAS Secretariat, email Atlas.Secretariat@cern.ch, FAX + 41 22 767 8350, to register as a member of ATLAS. See the CERN / ATLAS Registration information, and complete the required forms, i.e. the ATLAS Registration Form pdf and the CERN Registration Form available in English and French. If you have a valid Human Resource (HR) entry and an account at CERN then you do not need to sign the same agreement again. Please, however, get a different account for your ATLAS work.

- The ATLAS group code ("GG") is zp.
- You need an AFS/PLUS account to use the public Linux systems and to authorise access to some other ATLAS computing resources.
- You need a NICE/MAIL account to authorise access to some (network) services such as wireless access and the mailing list archives.

Note: Make sure you are properly registered with ATLAS Secretariat otherwise you will have problems. For instance you will not be able to use the mailing lists to ask for help and disk space etc.

2.1.2 Preparing your Account

You will be given a user name and a temporary password.

Now, log into a networked computer, launch a terminal and:

```
> ssh user_id@lxplus.cern.ch
```

Enter your temporary password when prompted and then a prompt will appear with the name of the CERN machine you are logged onto, and possibly your username. You should change your temporary password to something more memorable:

```
> kpasswd
```

and follow the instructions.

Now you should check your quota as follows:

```
> fs lq
```

You will see something like:

```
Volume.Name     Quota      Used      %Used      Partition
user.userID     50000      9734       19%          44%
```

The number underneath "quota" is in KB, and it is unlikely that you will have enough in your home directory for any serious work. You should therefore request more space from atlas-support@cern.ch [If the email bounces please read MailingListIssues]. Note that such space is designed to allow users of CERN machines the ability to perform some development, and to manage small test files. As such 500MB is the maximum that will generally be provided. Requests for space above this threshold will be treated as exceptional and will require additional justification. Requests from e.g. detector groups or physics working groups will be managed on a case by case basis.

Please note that you also may store files on CASTOR. For more information about that please look at New Castor pool

Naturally if you are using your own computer with AFS access you can probably use your local disk and won't need scratch space.

Finally, you may find it instructive to see where your home directory is in the greater scheme of the file system.

```
> pwd
```

This will appear:

```
/afs/cern.ch/user/<letter>/<userID>
```

The home directory will currently contain a few directories, including public. Your public directory and its contents is visible to any other AFS user. Everything else is restricted.

2.1.3 Information about your Account

To find out the logins and groups at CERN for user use the command:

```
> phone <userName> -A
```

The ATLAS group code is ZP and id 1307.

To find out all groups of user type:

```
> pts membership -name <userName>
```

Originally taken from SIT web pages and the static web page http://cern.ch/atlas-computing/documentation/faq.php.

Complete: 🟩
Responsible: Steve Lloyd
Author: Steve Lloyd 13 Apr 2005
Contributors: Steve Lloyd
Last significantly modified by: Steve Lloyd 28 Sep 2006
Last reviewed by: Steve Lloyd 25 Feb 2005

2.2 Setting up your Account

There are two different ways to use the ATLAS software:

- The AFS **Releases** - which are complete distibutions of the software
- The Pacman **Kits** - which don't include the source code

At an outside institute you may need to install the Kit (see WorkBookInstallingAtlasSoftware) unless you have direct AFS access to CERN or a mirror. (LCGG) Grid sites that support ATLAS software have the Kit installed.

If you use the Kits or the AFS Releases you have to prepare your account as described below first.

The instructions in this Workbook often contain instructions to execute scripts by doing something like

```
> source script_name.sh   [or .csh]
```

This means that if your login shell is **bsh**, **bash**, **ksh** or **zsh** etc you should use `script_name.sh` and if your login shell is **csh** or **tcsh** you should use `script_name.csh`. If you don't know what it is try typing `echo $SHELL`. The > represents the shell prompt and shouldn't be typed.

Note that due to limitations with the allowed length of `$SHELL`, **tcsh** is not currently supported for use with Athena. There is however an

"unofficial" workaround.

If you want to change your shell go to https://cra.cern.ch (you may have to do this in several places on the form).

2.2.1 Preparing your account to use the ATLAS software

ATLAS software is divided into packages, and these are managed through the use of a configuration management tool, CMT. This is used to copy ("check out") code from the main ATLAS repository and handle linking and compilation. If you are moving from BaBar you might be interested in BaBarToAtlas.

You must prepare your account, in the following way. Note that `cmthome` does not have to be in `$HOME`, it can be in any sub directory, but if so you will need to amend all the following examples accordingly.

```
> cd $HOME
> mkdir cmthome
> mkdir testarea
> mkdir testarea/14.2.23    (this is the directory in which you will work)
> cd cmthome
```

Then at CERN do:

```
> source /afs/cern.ch/sw/contrib/CMT/v1r20p20080222/mgr/setup.sh   [or .csh]
```

(go to http://cern.ch/atlas-computing/projects/releases/status/ and click on the link for a particular release to see what CMT version is needed)

or elsewhere do:

```
> source /path_to_kit/CMT/*/mgr/setup.sh [or .csh]
```

Where /path_to_kit/ is where the kit is installed (e.g. /opt/atlas/14.2.23).

The v1r20p20080222 string is the CMT version, which will change from time to time. Caution: there should be no directory $HOME/cmt, otherwise CMT will use the requirements file in this directory instead of using the one of $HOME/cmthome. If you add apply_tag setupCMT to the requirements file described below then one does not have to worry about getting the right version of CMT (one can use an old one to bootstrap).

If you are working at BNL, the set up is slightly different for 12.0.7 and r13 onward; these are kit releases. Please consult their Twiki for details.

Now using your text editor of choice, create a file called requirements. See AtlasLogin for a full explanation of what this does. If you are using a kit then you can copy a requirements file from /path_to_kit/cmtsite/requirements and customise it. See also UseAtlasSoftwareProjectsKit.

A basic requirements file at CERN would be:

```
#---------------------------------------------------------------------
set CMTSITE STANDALONE
set SITEROOT /afs/cern.ch/atlas/software/releases/14.2.23
macro ATLAS_TEST_AREA ${HOME}/testarea
macro ATLAS_DIST_AREA ${SITEROOT}
apply_tag projectArea
macro SITE_PROJECT_AREA ${SITEROOT}
macro EXTERNAL_PROJECT_AREA ${SITEROOT}
apply_tag opt
apply_tag setup
apply_tag simpleTest
use AtlasLogin AtlasLogin-* $(ATLAS_DIST_AREA)
set CMTCONFIG i686-slc4-gcc34-opt
#---------------------------------------------------------------------
```

If you are outside CERN you need to change SITEROOT accordingly:

```
set SITEROOT /path_to_kit
```

Now do:

```
> cmt config
```

You will only have to follow these procedures once until the version of CMT changes. Now kill the terminal window and log back in.

Note: These instructions set up CMT in so called 'User Mode' using release kits. Developers may wish to use a different set up known as 'Developer Mode' which uses the as-built releases and nightly builds on CERN AFS area (from the "builds" area), and the kit-built releases (from the "releases" area). See WorkBookAdvancedSetup#DevelopMode for more details.

Note: The `apply_tag setup` removes the necessity to `source /cmt/setup.[c]sh` for the packages that you check out. However if you do this then GANGA will not work in the same session.

Note: The `set CMTCONFIG` line effectively hardwires you to a particular platform (in this case 32 bit optimised) which cannot be overridden each time you start work (see below). Experienced users may wish to omit it and use the tags specified in AtlasLogin explicitly. ATLAS Releases 14 onwards only work with SL4.

Note: Make sure you have set `ATLAS_TEST_AREA` correctly in the `requirements` file or you will get errors saying that the include directory is not found in your work area when you try to compile for example. You will need to modify this requirements file if you change your test area directory. Also make sure you use `$HOME` or an absolute path to refer to your home directory and not `~`.

Note: The `requirements` files above use one default test area directory (`$HOME/testarea` or whatever you set). If you want the functionality to use any directory as a test area then you whould replace the `macro ATLAS_TEST_AREA` line with:

```
macro ATLAS_TEST_AREA "$PWD"
```

and then `cd` to the directory you want before doing `source ~/cmthome/setup.sh` as desribed below.

Note: By default the test directory has sub directories for each release e.g. `testarea/14.2.23`. If you include the line `apply_tag oneTest` then all releases use the same directory but you will need to clean it up if the release number changes. This might be useful if you follow the procedure above.

Some tips for using CMT can be found at SoftwareDevelopmentWorkbookCmtTips and more advanced setup instructions at WorkBookAdvancedSetup.

2.2.2 What to do every time you start work

If you are using a kit at a remote site and have AFS installed you should do kerberos authentication (before any ATLAS setup):

```
> kinit -5 username@CERN.CH    [Note CERN.CH in uppercase]
> klog username                [If you want to access your AFS files as well]
```

If you have problems with Kerberos then check your institute's firewall as described in `https://hypernews.cern.ch/HyperNews/Atlas/get/releaseKitProblem/77/1/1.html`. Additionally you may have to add

CERN.CH to the lists in /etc/krb.realms, /etc/krb.conf and /etc/krb5.conf (check e.g. the corresponding files on CERN's LXPLUS machines).

You need to issue the following command to set your terminal's environment variables to the correct values, and to set up CMT. You should do this every time you start a new terminal window in which you wish to use ATLAS software. With the requirements files above, you would type:

```
> source ~/cmthome/setup.sh -tag=14.2.23    [or .csh]
```

which would set the release number to 14.2.23 and the development area to $HOME/testarea/14.2.23

[If you did **not** specify apply_tag setup in your requirements file then you have to in addition type:

```
> source /afs/cern.ch/atlas/software/releases/14.2.23/AtlasOffline/14.2.23/
    AtlasOfflineRunTime/cmt/setup.sh
```

or the local equivalent path.]

You can add a tag "32" to make sure that the 32-bit version is used even on 64-bit machines (like CERN lxplus servers), as the 64-bit version of the athena software is not yet validated. However 32 is currently the default so this is not yet strictly necessary.

```
> source ~/cmthome/setup.sh -tag=14.2.23,32     [or .csh]
```

You might want to put these in your login script (.cshrc, .zshrc, .bashrc etc).

Note: If you want to use a **patched** production cache e.g. 14.2.23.3 instead of 14.2.23 you have to do:

```
source ~/cmthome/setup.sh -tag=AtlasProduction,14.2.23.3
```

and **Not** just

```
source ~/cmthome/setup.sh -tag=14.2.23
```

Note: With Release 14.2.23, you might get the following warnings which seems harmless and can be ignored:

```
#CMT> Warning: template <src_dir> not expected in pattern install_scripts (from
    TDAQCPolicy)
#CMT> Warning: template <files> not expected in pattern install_scripts (from
    TDAQCPolicy)
```

For more advanced setup options see WorkBookAdvancedSetup and AtlasLogin.

Testing your Setup

> echo $CMTCONFIG indicates the chosen directory name for compilation results (typically i686-slc4-gcc34-opt)

> echo $CMTPATH should list two paths: your own and the Atlas release. If not, see comments above regarding the requirements file. A third path with Gaudi is appended once you use your first package, and a fourth with LCGG common code. Note that it is possible to specify your own path CMTPATH by hand

: export CMTPATH=_ [or setenv CMTPATH] provided this is done after source ~/cmthome/setup.sh. This allows an easier switch between your working directories. But note that you should not try to switch releases by modifying directly CMTPATH.

> cmt should give you a list of cmt commands

> cmt show path lists areas where packages are taken in order of priority

Likely Problems

If later on you get errors like:

```
mkdir: cannot create directory '/include': Permission denied
```

then the setup probably failed. CMT should have appended your testarea to "/" but it was probably undefined and instead of /path_to_your_testarea/include you just have /include and you obviously don't have permission to create files in the root of the file system. You need to check the cmthome/requirements file and the source ~/cmthome/setup.sh command carefully. Things to check are:

- Make sure you have set ATLAS_TEST_AREA correctly
- Use $HOME$ not ∼
- Use $HOME/testarea not $HOME/testarea/14.2.23
- Don't source ~/cmthome/setup.sh more than once with different tags
- Make sure you are using the correct .sh or .csh for your shell
- Make sure you don't have an old .cmtrc file in your home directory
- Sometimes there are problems if you use symbolic links to get to your testarea.
  i.e. do cd $HOME/testarea/14.2.23/MyPackage rather than cd linktomypackage from somewhere else where linktomypackage is a symbolic link to $HOME/testarea/14.2.23/MyPackage.

If after using kinit, you subsequently have problems doing ssh e.g you get something like

```
/usr/bin/X11/xauth:  timeout in locking authority file /afs/cern.ch/user/l/lloyd
    /.Xauthority
hepix: E: /usr/bin/fs returned error, no tokens?
-bash: /afs/cern.ch/user/l/lloyd/.bash_profile: Permission denied
```

then do kinit -f -5 instead. This will allow ssh to forward your kerberos token. If that still fails try putting the following in your ~/.ssh/config file:

```
Host lxplus*
    GSSAPIDelegateCredentials yes
```

If you get errors like

```
athena.py: No such file or directory
```

then CMT probably didn't set up the run time environment properly. Make sure your cmthome/requirements has apply_tag setup in it or that you have -tag=14.2.23,setup when you do source ~/cmthome/setup.sh.

If all else fails start a new shell or terminal window and try again.

2.2.3 Using the Kits on the Grid

[This works on LCGG - what about US and Nordu?] Assuming you are running Athena from a shell script you need to include the following:

```
source $VO_ATLAS_SW_DIR/software/14.2.23/setup.sh
source $SITEROOT/AtlasOffline/14.2.23/AtlasOfflineRunTime/cmt/setup.sh
```

Previous Releases

- 13.0.30
- 12.0.6
- 11.0.5

Complete: 🟩
Responsible: Steve Lloyd
Author: James Catmore 03 Feb 2005
Contributors: James Catmore, Chris Collins Tooth, Steve Lloyd, David Rousseau
Last significantly modified by: Steve Lloyd 08 Feb 2008
Last reviewed by: Steve Lloyd 16 Oct 2008

2.3 Running Athena HelloWorld

This section describes how to run an example ATHENA algorithm (HelloWorld). This algorithm does little more than print some messages, but more complex algorithms are handled in exactly the same way.

Note: the location of the example algorithm in the code repository changed in release 15, see bottom of this page

Running HelloWorld

It is possible to run ATHENA algorithms without having the actual code in your development area. You will normally do this if you do not want to make any changes to the code - i.e. if you simply want to run the official released version. For this, several packages could be used such as the **TestRelease**, **RecExCommon** or **ControlTest** packages, can be used. These instructions explain how to run the HelloWorld algorithm using the **UserAnalysis** package.

First, log into your account and set up CMT (as described in WorkBookSetAccount) if you haven't already done so.

If you are using a kit at a remote site and have AFS installed you should do kerberos authentication:

```
> kinit -5 username@CERN.CH    [Note CERN.CH in uppercase]
> klog username                [If you want to access your AFS files as well]
```

Now do:

```
> source ~/cmthome/setup.sh -tag=14.2.23
[or source ~/cmthome/setup.csh -tag=14.2.23]
```

Now, go to your test area and check out the UserAnalysis package:

```
> cd testarea/14.2.23
> cmt co -r UserAnalysis-00-13-06 PhysicsAnalysis/AnalysisCommon/UserAnalysis
```

This will copy the UserAnalysis package, appropriate to this ATHENA release, from the ATLAS repository to your space. The co stands for "check out". Now, inspect the contents of the package.

(If you are using a kit and do not have AFS installed follow the instructions at WorkBookModAthenaHelloWorld#HelloWorldModKit).

```
> ls PhysicsAnalysis/AnalysisCommon/UserAnalysis
ChangeLog   UserAnalysis/          share/   src/   doc/   Root/   python/   CVS/
genConf/     i686-slc4-gcc34-opt/   cmt/     run/
```

Note: Prior to Release 12 there used to be another level in the directory structure i.e.
`PhysicsAnalysis/AnalysisCommon/UserAnalysis/UserAnalysis-NN-NN-NN`

Now configure and build the necessary files (this is not really necessary for just running HelloWorld unmodified):

```
> cd PhysicsAnalysis/AnalysisCommon/UserAnalysis/cmt
> source setup.sh    [or .csh]
> gmake
```

Now go to the `run` directory, get some Job Options and run:

```
> cd ../run
> get_files HelloWorldOptions.py
> athena.py HelloWorldOptions.py
```

The algorithm will first initialise and will then run ten times (during each run it will print various messages and echo the values given in the job options file). Then it will finalise, and stop. You should see something that includes this:

```
...
HelloWorld              INFO initialize()
HelloWorld              INFO   MyInt =    42
HelloWorld              INFO   MyBool =   1
HelloWorld              INFO   MyDouble = 3.14159
HelloWorld              INFO   MyStringVec[0] = Welcome
HelloWorld              INFO   MyStringVec[1] = to
HelloWorld              INFO   MyStringVec[2] = Athena
HelloWorld              INFO   MyStringVec[3] = Framework
HelloWorld              INFO   MyStringVec[4] = Tutorial
...
AthenaEventLoopMgr      INFO   ===>>>  start of run 0    <<<===
AthenaEventLoopMgr      INFO   ===>>>  start of event 1  <<<===
...
HelloWorld              INFO execute()
HelloWorld              INFO An INFO message
HelloWorld           WARNING A WARNING message
HelloWorld             ERROR An ERROR message
HelloWorld             FATAL A FATAL error message
...
```

If so you have successfully run Athena HelloWorld. The ERROR and FATAL error messages are just examples to show how you can issue such messages.

2.3.1 Likely problems

If you get errors like:

```
mkdir: cannot create directory '/include': Permission denied
```

initial setup probably failed. See WorkBookSetAccount#ProbLems for possible solutions.

If you get errors like:

```
"cvs checkout: GSSAPI authentication failed: Miscellaneous failure
cvs [checkout aborted]: GSSAPI authentication failed: Unknown code krb5 195
#CMT> The CVS pluggin is not installed or is not at protocol level v1r1
```

you probably do not have a Kerberos ticket which which to access CVS. Try doing `kinit` as described at the top even if you are not using a kit.

If you get errors like:

```
...
File "/afs/cern.ch/project/gd/LCG-share/3.0.10-1/glite/lib/python2.2/sre_compile
    .py",
  line 17, in ?
>>      assert _sre.MAGIC == MAGIC, "SRE module mismatch"
>> AssertionError: SRE module mismatch
```

you probably sourced the wrong version of the Grid UI. See WorkBookStartingGrid#UserInterface or don't source the UI unless you have to.

Running HelloWorld on the Lxbatch

The public logon machines (Lxplus) are intended as interactive service for all CERN users, therefore long and CPU instensive jobs should be sent to the Batch system (they get killed automatically on Lxplus). Although this is not necessary for short jobs like HelloWorld it will be necessary for long tasks like running simulation.

First prepare a small job script, such as `myjob` and make it executable (`chmod +x myjob`):

```
#!/bin/bash
source ~/cmthome/setup.sh -tag=14.2.23
cd ~/testarea/14.2.23/PhysicsAnalysis/AnalysisCommon/UserAnalysis/run
athena.py HelloWorldOptions.py
```

Now you can submit it to the batch system via:

```
> bsub -q 1nh myjob
```

The `-q` switch selects the job queue which can be `8nm` (8 minutes), `1nh` (1 hour), `8nh` (8 hours), `1nd` (1 day) and `1nw` (1 week). The default is `8nm`. (The `n` stands for normalised). To select certain resources (i.e. define minimum requirements for your job to start) use the `-R` option, e.g. to select at least 500 MB in /tmp, 200 MB free memory, 400 MB swap and 1 GB pool disk space:

```
> bsub -R "tmp>500&&mem>200&&swp>400&&pool>1000" myjob
```

You can check the status of your job(s) by executing the `bjobs` command:

```
> bjobs
```

which also takes a `jobID` as parameter for the "long format" with detailed information:

```
> bjobs -l jobID
```

You can kill your jobs using `bkill`:

```
> bkill jobID
```

Running HelloWorld in Release 15

In release 15 the HelloWorld example lives in `Control/AthenaExamples/AthExHelloWorld` in AtlasCore.

To run the example, set up the Athena environment, then issue

```
athena.py AthExHelloWorld/HelloWorldOptions.py
```

Previous Releases

- 13.0.30
- 12.0.6
- 11.0.5

Complete: 🟩
Responsible: Steve Lloyd
Author: James Catmore 03 Feb 2005
Contributors: James Catmore, Chris Collins Tooth, Markus Cristinziani, Steve Lloyd, Oxana Smirnova
Last significantly modified by: Steve Lloyd 08 Feb 2008
Last reviewed by: Steve Lloyd 16 Oct 2008

2.4 Modifying Athena HelloWorld

This section describes how to modify the example Athena algorithm (HelloWorld) described in WorkBookRunAthenaHelloWorld.

Modifying HelloWorld with the Release

If you are using a kit and do not have AFS installed follow the instructions at WorkBookModAthenaHelloWorld#HelloWorldModKit below first.

Go to your testarea and check out the **AthExHelloWorld** package:

```
> cd testarea/14.2.23
> cmt co -r AthExHelloWorld-01-01-01 Control/AthenaExamples/AthExHelloWorld
```

Navigate to the `src` directory

```
> cd Control/AthenaExamples/AthExHelloWorld/src
```

and edit the C++ file `HelloAlg.cxx` for instance by adding this to the initialize() method:

```
log << MSG::INFO << "   Hello ATLAS User" << endreq;
```

and save the file.

Now build this package:

```
> cd ../cmt
> cmt config
> source setup.sh  [or .csh]
> gmake
```

Go back to the **UserAnalysis** package and rerun Athena:

```
> cd ../../../../PhysicsAnalysis/AnalysisCommon/UserAnalysis/run
> athena.py HelloWorldOptions.py
```

You should now see this somewhere in the output:

```
HelloWorld              INFO    Hello ATLAS User
```

2.4.1 Cleaning Up

If you want to cleanup and start again without deleting everything, you can do:

```
gmake clean
```

However, this may not quite get rid of everything so you might like to do this from the `/cmt` directory:

```
cmt rm -fR ../i686-slc4-gcc34*
```

and this from the `testarea/14.2.23` area:

```
rm -fr InstallArea
```

Modifying HelloWorld with a Kit

If you have AFS installed you can check out the packages directly and then follow the instructions above.

If not then you can log in to CERN, go to your test directory, check out the packages, and then make a tar file to bring back:

```
> tar -cvlpf checkout.tar Control PhysicsAnalysis
```

Then on your home computer:

```
> scp lxplus.cern.ch:testarea/14.2.23/checkout.tar .
> tar -xf checkout.tar
```

and proceed as above.

Modifying HelloWorld on the LCGG Grid

There are several ways to do this. One way is to make a tar of the `Control` package as shown above and extract and modify the `HelloAlg.cxx` file.

Then modify the `hello.jdl` file to include these in the input sandbox:

```
############# Athena #################
Executable = "hello.sh";
StdOutput = "hello.out";
StdError = "hello.err";
InputSandbox = {"hello.sh","AthExHelloWorld_jobOptions.py","checkout.tar","
    HelloAlg.cxx"};
OutputSandbox = {"hello.out","hello.err", "CLIDDBout.txt"};
Requirements = Member("VO-atlas-offline-14.2.23-i686-slc4-gcc34-opt", other.
    GlueHostApplicationSoftwareRunTimeEnvironment);
###############################################
```

and to modify the `hello.sh` script thus:

```
#!/bin/bash
# Script to run AthenaHelloWorld on the Grid

source $VO_ATLAS_SW_DIR/software/14.2.23/setup.sh
source $SITEROOT/AtlasOffline/14.2.23/AtlasOfflineRunTime/cmt/setup.sh

export CMTPATH=`pwd`:${CMTPATH}

# Create the checked out package then replace the modified file

tar -xf checkout.tar

cp HelloAlg.cxx Control/AthenaExamples/AthExHelloWorld/src/

# Rebuild

cd Control/AthenaExamples/AthExHelloWorld/cmt

cmt config
source setup.sh
gmake

cd ../../../../PhysicsAnalysis/AnalysisCommon/UserAnalysis/run

athena.py HelloWorldOptions.py
```

Previous Releases

- 13.0.40
- 12.0.6
- 11.0.5

Complete: 🟩

Responsible: Steve Lloyd
Author: Steve Lloyd 12 Apr 2005
Contributors: Steve Lloyd
Last significantly modified by: Steve Lloyd 08 Feb 2008
Last reviewed by: Steve Lloyd 16 Oct 2008

2.5 Creating a New Package

This section demonstrates the construction of a new ATHENA package (containing a single algorithm). There are two steps - the creation of the empty package, and the insertion of the algorithm. We assume the package is called **MyNewPackage** and the algorithm **MyAlg**. If you want to check your new code into CVS you should consult the Software Development Workbook.

Creating a New Package

First, log into your account and set up CMT (as described in WorkBookSetAccount):

Now, go to your test area and create a new package:

```
> cd testarea/14.2.23
> cmt create MyNewPackage MyNewPackage-00-00-01
```

If your package is in a subdirectory such as `MyPath/MySubPath` put this path as an extra argument at the end of the `cmt create` command.

```
> cd testarea/14.2.23
> cmt create MyNewPackage MyNewPackage-00-00-01 MyPath/MySubPath
```

Do **not** put `MyPath/MySubPath/MyNewPackage`.

Go to the `cmt` directory of your new package:

```
> cd MyNewPackage/cmt
```

and create (or replace) a file called `requirements`. The requirements file must contain the following:

- The package name
- The author name
- The package dependencies (i.e. a list of other packages needed by your package)
- A list of all the source code used by the package

A minimal `requirements` file would be:

```
#####################################################
package MyNewPackage
author ATLAS Workbook
use AtlasPolicy AtlasPolicy-01-*
use GaudiInterface GaudiInterface-01-* External
library MyNewPackage *.cxx -s=components *.cxx
apply_pattern component_library
apply_pattern declare_joboptions files="MyJobOptions.py"
#####################################################
```

Inserting the algorithm

The algorithm source code goes into the src directory of the package, and the header file into the directory with the name of the package. The example algorithm provided does little other than produce a single message; its purpose is simply to demonstrate how to write new algorithms.

```
> cd ../src
```

Create a file MyAlg.cxx:

```
#include "MyNewPackage/MyAlg.h"
#include "GaudiKernel/MsgStream.h"
//////////////////////////////////////////////////////////////////////////////
MyAlg::MyAlg(const std::string& name, ISvcLocator* pSvcLocator) :
Algorithm(name, pSvcLocator)
{
  // Properties go here
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
StatusCode MyAlg::initialize(){
    MsgStream log(msgSvc(), name());
    log << MSG::INFO << "initialize()" << endreq;
    return StatusCode::SUCCESS;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
StatusCode MyAlg::execute() {
    MsgStream log(msgSvc(), name());
    log << MSG::INFO << "execute()" << endreq;
    log << MSG::INFO << "Your new package and algorithm are successfully
        installed" << endreq;

    return StatusCode::SUCCESS;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
StatusCode MyAlg::finalize() {
    MsgStream log(msgSvc(), name());
    log << MSG::INFO << "finalize()" << endreq;
    return StatusCode::SUCCESS;
}
```

You also need to create two files that are used to create entry points into a dynamically loaded library, so you can say at runtime "load this alg" and it will be found in the correct library and be loaded into memory.

```
> mkdir components
> cd components
```

Create a file called MyNewPackage_entries.cxx containing this:

```
#include "MyNewPackage/MyAlg.h"
#include "GaudiKernel/DeclareFactoryEntries.h"
DECLARE_ALGORITHM_FACTORY( MyAlg )
DECLARE_FACTORY_ENTRIES(MyNewPackage) {
DECLARE_ALGORITHM( MyAlg )
}
```

and a file called MyNewPackage_load.cxx containing this:

```
#include "GaudiKernel/LoadFactoryEntries.h"
LOAD_FACTORY_ENTRIES(MyNewPackage)
```

Next you should create the header files:

```
> cd ../..
> mkdir MyNewPackage
> cd MyNewPackage
```

Create a file MyAlg.h:

```
#include "GaudiKernel/Algorithm.h"
/////////////////////////////////////////////////////////////////////////////
 class MyAlg:public Algorithm {
    public:
    MyAlg (const std::string& name, ISvcLocator* pSvcLocator);
    StatusCode initialize();
    StatusCode execute();
    StatusCode finalize();
 };
```

Finally, you should create a JobOptions file.

```
> cd ..
> mkdir share
> cd share
```

Create a file MyJobOptions.py:

```
#--------------------------------------------------------------
# Private Application Configuration options
#--------------------------------------------------------------
# Full job is a list of algorithms
from AthenaCommon.AlgSequence import AlgSequence
job = AlgSequence()
# Add top algorithms to be run
from MyNewPackage.MyNewPackageConf import MyAlg
job += MyAlg( "MyNewPackage" )   # 1 alg, named "MyNewPackage"
#--------------------------------------------------------------
# Set output level threshold (DEBUG, INFO, WARNING, ERROR, FATAL)
#--------------------------------------------------------------
job.MyNewPackage.OutputLevel = INFO
#--------------------------------------------------------------
# Event related parameters
#--------------------------------------------------------------
# Number of events to be processed (default is 10)
theApp.EvtMax = 1
#==============================================================
```

Note that you do not need to create MyNewPackageConf.py yourself. This is generated as part of the build process in the genconf directory of MyNewPackage and then installed in InstallArea/python/MyNewPackage. Do *not* name the JobOptions file the same as your package, meaning MyNewPackage.py, since this will give an error message and cause Athena to crash.

Building the New Package

Now build your package:

```
> cd ../cmt
> cmt config
> source setup.sh
> gmake
```

If all is well you should be able to go to your run directory:

```
> cd ~/testarea/14.2.23
> cd PhysicsAnalysis/AnalysisCommon/UserAnalysis/run
> cp  ../../../../MyNewPackage/share/MyJobOptions.py .

> athena.py MyJobOptions.py
```

If all is well you should see lines like this in the output:

```
MyAlg          INFO initialize()
MyAlg          INFO execute()
MyAlg          INFO Your new package and algorithm are successfully installed
MyAlg          INFO finalize()
```

Previous Releases

- 13.0.30
- 12.0.6
- 11.0.5

Originally taken from http://jcatmore.home.cern.ch/jcatmore/NESCNewAlgs.htm (inactive URL)

Complete: 🟩
Responsible: Steve Lloyd
Author: James Catmore 03 Feb 2005
Contributors: James Catmore, Chris Collins Tooth, Steve Lloyd
Last significantly modified by: Steve Lloyd 08 Feb 2008
Last reviewed by: Steve Lloyd 16 Oct 2008

2.6 More Advanced Setup Options

2.6.1 User versus Developer Mode

The setup instructions described in WorkBookSetAccount#UserMode are recommenced for users (hence "User Mode"). This is the recommended procedure for dealing with a kit installed release at any site (where CERN is just another site, nothing special). You specify the location of the release, which in this case is unique since every release is installed in a separate directory, and optionally you specify the platform if multiple platforms are installed alongside each other at the same location. If only a single platform is installed, you don't need to specify the platform, or you can accept the default platform (which is 32-bit - which is why you don't need to specify the "32" tag). No other tags need to be specified, and unfortunately some of them (such as the "releases" tag) cause problems in the setup and shouldn't be specified. This access mode is mainly designed for users of the software - people just running against a stable numbered release, or at most doing restricted development of a few packages for physics analysis against a stable numbered release. Hence we refer to it as "User" Mode.

However one can replace the relevant lines in the `requirements` file with:

```
set CMTSITE CERN
set SITEROOT /afs/cern.ch
macro ATLAS_DIST_AREA ${SITEROOT}/atlas/software/dist
apply_tag 32            # Makes sure you use 32-bit versions
apply_tag setup         # setup the run time environment
```

In contrast with "User" mode, this is a model which allows you to access both the as-built releases and nightly builds on CERN AFS (from the "builds" area), and the kit-built releases (from the "releases" area). You have flexibility in accessing any release in either location, the nightly builds, different platforms, etc. It's designed primarily for developers who are primarily running against the nightly releases, developing code which is a candidate for incorporation into a subsequent numbered release. it also provides support for help desk support people trying to reproduce problems reported by others and wishing to reproduce a large variety of setup configurations. Hence we refer to it as "Developer" Mode.

The setup for both modes of operation is driven by the AtlasLogin package which has built-in knowledge of the disk layout at CERN plus the ability to accept local site-specific overrides for usage at other sites in Developer Mode. The only required bootstraps are the `SITEROOT` environment variable which all other directory locations are relative to, the `CMTSITE` environment variable (which can essentially be either `CERN` or `STANDALONE`) and the `ATLAS_DIST_AREA` macro which specifies the location of the AtlasLogin package itself. Setting `CMTSITE` to `STANDALONE` indicates that *only* kit-built releases are available at the location specified by `SITEROOT`. Setting `CMTSITE to CERN` indicates that this is a CERN-like site and uses by default the built-in knowledge embedded within AtlasLogin itself (corresponding to CERN itself). Perhaps confusingly, a remote site could be configured to behave similar to CERN, with nightly builds, as-built releases and kit-built releases, by not only setting `CMTSITE` to `CERN` but also specifying some other macros and tags to override the built-in CERN locations.

The location of the actual files at CERN is:

$ `/afs/cern.ch/atlas/software/releases` is the location for the kit-built releases at CERN, installed one per directory. This location is accessed at CERN by using the "releases" tag in order to override the built-in default.

$ `/afs/cern.ch/atlas/software/dist` is the location of the AtlasLogin package at CERN (and happens to contain also some old monolithic releases).

$ `/afs/cern.ch/atlas/software/builds` is the location for the as-built releases at CERN, all alongside each other rather than one per directory. The nightly builds are located in the `nightlies/` directory within this location. This location is accessed at CERN by using the "builds" tag, but this is also the CERN default location so doesn't need to be specified.

In "Developer Mode" one needs to add the "32" tag and "releases" tag for patched caches.

```
> source ~/cmthome/setup.sh -tag=14.2.23,32     [or .csh]
> source ~/cmthome/setup.sh -tag=AtlasProduction,14.2.23.3,32,releases
```

2.6.2 Change the Build Mode (debug, optimize, etc)

This assumes you already have set up a cmthome directory as described in WorkBookSetAccount. Do

```
source ~/cmthome/setup.sh -tag_add=dbg or -tag_add=opt  or -tag_add=prof
```

Check that CMTPATH and CMTCONFIG are what you want and then setup and rebuild (no clean-up necessary):

```
source ./setup.sh
gmake
```

or, if several packages have been checked out in the same user area:

```
source ./setup.sh
cmt broadcast gmake
```

-#CleanupBinaries –++ Cleanup the Binaries

From the cmt directory do:

```
rm -fR ../i686*
```

or even better, since more generic

```
gmake binclean
```

or, if several packages have been checked out in the same user area:

```
cmt broadcast rm -fR ../i686*
```

where i686* is the directory where the binaries are being built (CMTCONFIG).

```
cmt broadcast gmake binclean
```

2.6.3 Change the Release

First select the new release the usual way (generally done by editing your ~/cmthome/cmt requirements file):

```
source ~/cmthome/setup.sh
```

Update the environment variables (assuming you are in the cmt directory of a package, e.g. TestRelease):

```
source setup.sh
```

Then clean-up the binaries (as above) and rebuild everything:

```
gmake
```

or, if several packages have been checked out in the same user area:

```
cmt broadcast gmake
```

If you have checked out and modified some packages from CVS, before rebuilding you need to update each package to its possible new version in the new release. To do this, the recommended way is to do the following. cd to the top directory of the package. Find the new tag of the package in the release (described elsewhere). Then to see what has changed:

```
cvs diff -r < new package tag >
```

To merge your own changes with the packages changes:

```
cvs update -r < new package tag >
```

Watch out for message like "Conflicts". This means that your modification and the package modifications are in conflicts, typically the same lines have been edited. The files were this happens need be edited, area of conflicts are indicated with ">>>>>>>>>>>>.". Do not forget to update jobOptions files in /run directory if necessary.

2.6.4 Change a Compiler Option

Let's suppose you want to use -pedantic-errors when compiling a given package. Edit the cmt/requirements file. Add the following lines:

```
private
macro_append cppflags " -pedantic-errors"
```

(be careful not to forget the leading space). The new compiler option will be used whenever a file of this package is recompiled. If private: is omitted, the new compiler option will be used also in any package which depend on it directly or indirectly.

2.6.5 Select a Work Area

A work area is a directory where you check out a package and work on it. If only one package is considered, you have nothing to do to qualify a directory as being a work area. Thus for such a simple scenario you may skip this section.

However, when several packages are expected to be developed in a given work area, CMT needs to know it, so as to include it in the package search path list (this is required to identify where to search used packages). The prefered method is to put

```
macro ATLAS_TEST_AREA "$PWD"
```

in your cmthome/requirements file as described in WorkBookSetAccount.

Another useful thing is to setup a dummy package with a "use" to all checked out packages. Then from this package one can use cmt broadcast ... to broadcast commands to each package. Where most typically one would do cmt broadcast gmake. To set up this dummy package (called WorkArea), do:

```
cd <work area>
setupWorkArea.py
```

then one can:

```
cd WorkArea/cmt
cmt broadcast gmake
```

Note that any command can be broadcast. For example, to run `cvs -n update` in each package one does:

```
cmt broadcast - "cd ..; cvs -n update"
```

where the '-' means continue even if there is an error in one of the packages, and the command is to cd to the directory above 'cmt' and run the 'cvs -n update'.

2.6.6 Use of several cmthome/requirements files

The current page and the page WorkBookSetAccount describe the use of `cmthome/requirements` for the case that this directory is located in `$HOME`. But the directory `cmthome` can be located in any directory and several different cmthome/requirements files can exist in parallel in different directories. This is for example useful, when one works on several releases, or if one works on different projects with different settings. To switch from one to the other, you may use tags, as described above, or you may set up two different requirements files like in this example: Assume you have defined requirements files for different releases

```
~/testarea/13.0.10/cmthome/requirements
~/testarea/14.2.0/cmthome/requirements
```

The requirements files should follow the examples provided in WorkBookSetAccount. Now you may configure and source the environment, which you intend to work with, e.g. 14.2.0

```
cd  ~/testarea/14.2.0/cmthome
cmt config
source setup.sh     [or.csh]
```

This is often simpler then the use of tags. When you change from one setting to the other, it is best to restart in a new terminal to ensure that the CMT environment does not contain any previous definitions. Note: The command `cmt config` is needed only once, or when the version of CMT changes.

2.6.7 List of all tags for use in cmthome/requirements

See the page AtlasLogin for a list and explanation of all tags, which can be used in the `cmthome/requirements`.

Some of this information was originally in InformationForUsers.

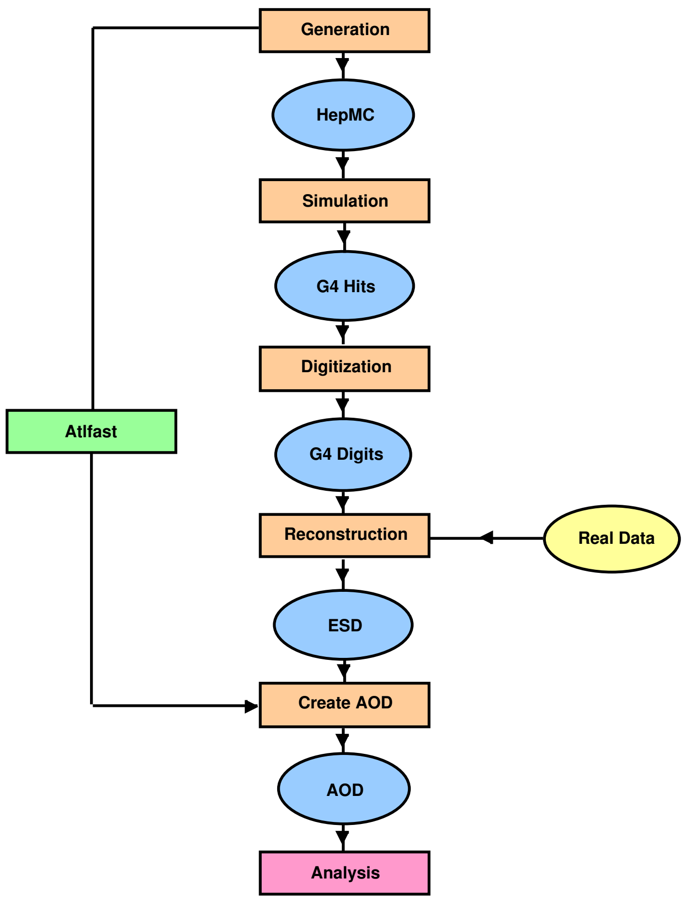
Complete: ▨
Responsible: Steve Lloyd
Author: Edward Moyse 08 Feb 2005
Contributors: Steve Lloyd, Edward Moyse, David Rousseau
Last significantly modified by: Steve Lloyd 25 Jan 2008
Not yet reviewed

Chapter 3

Running Athena

3.1 Running the Full Chain

In order to produce Monte Carlo events on which to perform analysis a **Full Chain** of steps needs to be taken from Generation to production of Analysis Object Data (AOD) as shown in the diagram. Because of the time this process takes, especially the Simulation stage, it is unlikely that most users will produce many events themselves but will rely on centrally produced events. However limited test samples of specific channels may be necessary from time to time. Each of these stages is described separately but they can be combined if required.



Schematic representation of the Full Chain Monte Carlo production.

You can short circuit the Full Chain by using **Atlfast** which provides a fast simulation of the whole chain by taking the generated events and smearing them to produce AOD directly. Atlfast can in fact take input from any of the event generator, simulation, digitization, or Event Summary Data (ESD) files.

Database Errors

If you are using a remote kit older than 10.4.0 and you get database errors such as

```
RDBAccessSvc::c...  ERROR Could not connect to the database server.
```

You should add (or replace) these lines in your job options:

```
RDBAccessSvc = Service( "RDBAccessSvc" )
RDBAccessSvc.HostName   = "atlas_dd"
```

and replace the local copy of

```
/afs/cern.ch/project/oracle/admin/tnsnames.ora
```

with the one from CERN.

Complete: 🟨🟨⬜
Responsible: Steve Lloyd
Author: Steve Lloyd 27 Apr 2005
Contributors: Steve Lloyd, Pete Watkins
Last significantly modified by: Steve Lloyd 18 Jan 2006
Last reviewed by: Pete Watkins 25 Jul 2005

3.2 Generating Events with Athena

This section describes how to generate events with Athena. Generation refers to the production of particle four vectors from specified physics processes. There are many event generators available. See AtlasProtected.McGeneratorsForAtlas for further information. The examples given here use **Pythia**. For information about the various Pythia options that can be set you may need to consult the Pythia Manual and other information on AtlasProtected.PythiaForAtlas.

You first need to set up your account as described in WorkBookSetAccount and check out a package to work in such as **UserAnalysis** as described in WorkBookRunAthenaHelloWorld.

3.2.1 Running Pythia

Go to your run directory such as:

```
> cd testarea/14.2.23
> cd PhysicsAnalysis/AnalysisCommon/UserAnalysis/run
```

You need to obtain the following two files via:

```
> get_files PDGTABLE.MeV
> get_files jobOptions.pythia.py
```

You can now run Athena with these job options. Because there will be a lot of output it is best to direct it to a file.

```
> athena.py jobOptions.pythia.py > athena_gen.out
```

If you want to have the output on the screen as well as in the file do:

```
> athena.py jobOptions.pythia.py | tee athena_gen.out
```

You can now inspect the `athena_gen.out` file which should have the full event output from 5 events. The type of events generated and Pythia options are set in the job options by the line:

```
Pythia.PythiaCommand = ["pysubs msel 13","pysubs ckin 3 18.","pypars mstp 43 2"]
```

- MSEL = 13 selects Z0 + Jet Production
- CKIN(3) = 18. sets the minimum PT for hard processes in GeV
- MSTP(43) = 2 sets Z0/gamma* interference off (only Z0 included)

3.2.2 Generating Z0→e+e-

Now modify the Pythia options to generate Z0→e+e- events only. Either edit the file `jobOptions.pythia.py` or create a file `myGenOptions.py` containing the following:

```
#--------------------------------------------------------------
# Private Application Configuration option
#--------------------------------------------------------------
from AthenaCommon.AppMgr import ServiceMgr
ServiceMgr.MessageSvc.OutputLevel = INFO
#--------------------------------------------------------------
# Event related parameters
#--------------------------------------------------------------
# Number of events to be processed (default is 10)
theApp.EvtMax = 10
#--------------------------------------------------------------
# Algorithms Private Options
#--------------------------------------------------------------
from AthenaServices.AthenaServicesConf import AtRndmGenSvc
ServiceMgr += AtRndmGenSvc()
ServiceMgr.AtRndmGenSvc.Seeds = ["PYTHIA 4789899 989240512",
"PYTHIA_INIT 820021 2347532"]
#
from AthenaCommon.AlgSequence import AlgSequence
job=AlgSequence()
#
from Pythia_i.Pythia_iConf import Pythia
job +=Pythia()
# Generate Z->ee
job.Pythia.PythiaCommand = ["pysubs msel 0","pysubs msub 1 1",
                            "pypars mstp 43 2","pydat3 mdme 174 1 0",
                            "pydat3 mdme 175 1 0","pydat3 mdme 176 1 0",
                            "pydat3 mdme 177 1 0","pydat3 mdme 178 1 0",
                            "pydat3 mdme 179 1 0","pydat3 mdme 180 1 0",
                            "pydat3 mdme 181 1 0","pydat3 mdme 182 1 1",
                            "pydat3 mdme 183 1 0","pydat3 mdme 184 1 0",
                            "pydat3 mdme 185 1 0","pydat3 mdme 186 1 0",
                          "pydat3 mdme 187 1 0"]

from TruthExamples.TruthExamplesConf import DumpMC
job += DumpMC()
#--------------------------------------------------------------------------
# Pool Persistency
#--------------------------------------------------------------------------
from AthenaPoolCnvSvc.WriteAthenaPool import AthenaPoolOutputStream
Stream1 = AthenaPoolOutputStream( "Stream1" )
Stream1.OutputFile = "pythia.pool.root"
Stream1.ItemList += [ 'EventInfo#*', 'McEventCollection#*' ]
###########################################################################
```

The `Stream1.OutputFile` line specifies the output file name (`pythia.pool.root`) which you should modify appropriately.

The `Pythia.PythiaCommand` lines switch off all processes, turns on Z0 production, turns off Z0/gamma* interference and switches off all Z0 decay modes except Z0→e+e-. (To get Z0→mu+mu- only, change `"pydat3 mdme 182 1 1"` to `"pydat3 mdme 182 1 0"`, and change `"pydat3 mdme 184 1 0"` to `"pydat3 mdme 184 1 1"`.)

You might also want to change the output level threshold (`MessageSvc.OutputLevel`) to cut down on the printout. To stop the dumping of events to the output remove the `DumpMC` from `theApp.TopAlg`.

If you are preparing a run that has an official run number (let's take 999999 as an example), add the following to your jobOptions:

```
import AthenaCommon.AtlasUnixGeneratorJob
svcMgr.EventSelector.RunNumber = 999999
```

Don't forget to modify the seeds (`AtRndmGenSvc.Seeds`) for each new run ... else you will generate the same events again and again !

You can now rerun Athena using

```
> athena.py myGenOptions.py > athena_gen.out
```

The output should contain vertices with an incoming Z0 (ID 23) and outgoing e+ (ID -11) and e- (ID 11) plus possibly extra photons (ID 22). You should have a file of generated events e.g.

```
-rw-r--r--  1 lloyd lloyd    254185 May 23 16:48 pythia.pool.root
```

The above example is for educational purposes only (the underlying event is not as it will be in Atlas), see AtlasProtected.GeneratorValidation for details about settings for events that are useful for physics analyis. Further information is also available at the generators tutorial.

Previous Releases

- 13.0.30
- 12.0.6

Complete: 🟩
Responsible: Steve Lloyd
Author: James Catmore 03 Feb 2005
Contributors: James Catmore, Chris Collins Tooth, Steve Lloyd
Last significantly modified by: Steve Lloyd 08 Feb 2008
Last reviewed by: Steve Lloyd 16 Oct 2008

3.3 Simulating Events with Athena

This section describes how to simulate previously generated events with Athena. Simulation is the process whereby generated events are passed through a GEANT4 Simulation of the ATLAS Detector to produce GEANT4 **Hits** i.e. a record of where each particle traversed the detector and how much energy etc was deposited.

You first need to set up your account as described in WorkBookSetAccount and check out a package to work in such as **UserAnalysis** as described in WorkBookRunAthenaHelloWorld. The generation of events is described in WorkBookGeneration.

3.3.1 Running Simulation in Athena

The simulation interface is different than you will see in other parts of the chain. The reason for this is that it allows direct access to Geant4 through the python interface. For full details, see the official simulation documentation.

Note If you are using Release 14.2.20 you need to check out `Event/EventInfoMgt-00-01-22` otherwise the Digitization step will not work. This is fixed in 14.2.23.

```
> cd testarea/14.2.20
> cmt co -r EventInfoMgt-00-01-22 Event/EventInfoMgt
> cd Event/EventInfoMgt/cmt
> source setup.sh ""
> gmake
> cd ../../../../..
```

Go to your run directory such as:

```
> cd testarea/14.2.23
> cd PhysicsAnalysis/AnalysisCommon/UserAnalysis/run
```

You need to obtain the following file via:

```
> get_files jobOptions.G4Atlas_Sim.py
```

If you haven't already got it from the generation step you may have to:

```
> get_files PDGTABLE.MeV
```

Change the following lines in `jobOptions.G4Atlas_Sim.py` appropriately:

```
SimFlags.SimLayout='ATLAS-CSC-02-02-00'
athenaCommonFlags.PoolHitsOutput='g4hits.pool.root'
```

Comment out the following:

```
# - uses single particle generator
#SimFlags.KinematicsMode='SingleParticle'
#SimFlags.ParticlePDG='11'
#   set energy constant to 10 GeV
#SimFlags.Energy=10000
#  sets the EtaPhi, VertexSpread and VertexRange checks on
#SimFlags.EventFilter.set_On()
```

Uncomment the following and modify appropriately:

```
SimFlags.KinematicsMode='ReadGeneratedEvents'
athenaCommonFlags.PoolEvgenInput=['pythia.pool.root']
```

Modify the following appropriately:

```
athenaCommonFlags.EvtMax=10
```

Note that due to a bug if you set `theApp.EvtMax` to -1, then `G4Atlas_Sim.py` will override it. **Note** that if you use !SkipEvents(), `G4Atlas_Sim.py` will override it as well!! In that case, you need to call `EventSelector.SkipEvents()` after `include('G4Atlas_Sim.py')`.

You can now run athena:

```
> athena.py  jobOptions.G4Atlas_Sim.py > athena_sim.out 2>&1
```

Simulation takes a LONG time (~15 mins/event for Z0→e+e-). On Lxplus your command will be killed after 40 mins so it is better to use Lxbatch as described here.

Eventually you should have a file of Geant4 Hits (this was 10 Z0→e+e- events):

```
-rw-r--r--  1 lloyd lloyd  12996310 May 23 20:02 g4hits.pool.root
-rw-r--r--  1 lloyd lloyd    254185 May 23 16:48 pythia.pool.root
```

Note: If you get errors like this:

```
DllClassManager     ERROR Could not load module Geo2G4
DllClassManager     ERROR System Error: libG4global.so: cannot open shared
```

make sure you have set up the run time environment properly as described in WorkBookSetAccount#ProbLems.

Simulation Layout/Detector Description

The line `SimFlags.SimLayout.set_Value('ATLAS-CSC-02-02-00')` defines the version of the Simulation Layout (also known as the Detector Description) to use. Information on possible values can be found at AtlasGeomDBTags#ATLAS_Geom_DB_Tag_contents.

Previous Releases

- 13.0.30
- 12.0.6
- 11.0.5

- 10.0.1

Complete: 🟩
Responsible: Steve Lloyd
Author: James Catmore 03 Feb 2005
Contributors: James Catmore, Chris Collins Tooth, Gernot Krobath, Steve Lloyd
Last significantly modified by: Steve Lloyd 08 Feb 2008
Last reviewed by: Steve Lloyd 16 Oct 2008

3.4 Digitizing Events with Athena

This section describes how to digitize previously simulated events with Athena. Digitization is the process whereby the GEANT4 **Hits** from the simulation are subjected to the response of the detector to produce **Digits**, such as times and voltages, as produced in the Raw Data from the real detector.

You first need to set up your account as described in WorkBookSetAccount and check out a package to work in such as **UserAnalysis** as described in WorkBookRunAthenaHelloWorld. The simulation of events is described in WorkBookSimulation.

3.4.1 Running Digitization in Athena

Go to your run directory such as:

```
> cd testarea/14.2.23
> cd PhysicsAnalysis/AnalysisCommon/UserAnalysis/run
```

Create a small job options file such as `myDigiOptions.py` with the following lines in it, with the input and output files set appropriately:

```
###############################################################
# User Digitization Job Options

from AthenaCommon.AthenaCommonFlags import jobproperties
jobproperties.AthenaCommonFlags.EvtMax = -1
jobproperties.AthenaCommonFlags.PoolHitsInput=["g4hits.pool.root"]
jobproperties.AthenaCommonFlags.PoolRDOOutput="g4digi.pool.root"

# Detector description
from AthenaCommon.GlobalFlags import jobproperties
jobproperties.Global.DetDescrVersion='ATLAS-CSC-02-02-00'

###############################################################
```

The line `jobproperties.Global.DetDescrVersion`'ATLAS-CSC-02-02-00'= defines the version of the Detector Description to use as described in AtlasGeomDBTags#ATLAS_Geom_DB_Tag_contents.

You can now run athena:

```
athena.py myDigiOptions.py Digitization/Digitization.py > athena_digi.out
```

You should have a file of Geant4 Digits (this was 10 Z0→e+e- events):

```
-rw-r--r--  1 lloyd lloyd  19172652 May 23 20:05 g4digi.pool.root
-rw-r--r--  1 lloyd lloyd  12996310 May 23 20:02 g4hits.pool.root
-rw-r--r--  1 lloyd lloyd    254185 May 23 16:48 pythia.pool.root
```

Be careful that you have enough disk space! In the release 11.0.x series, the digitization adds noise to the calorimeter (see CaloDigitization), resulting in very large output files. Further details are available in AtlasDigitization.

Note If you are using Release 14.2.20 and you get errors like this:

```
IOVDbSvc             FATAL No IOVDbSvc.GlobalTag set on job options or input file
    .
                                 Conditions data to use is UNDEFINED so STOP
```

You need to check out `Event/EventInfoMgt-00-01-22` during the Simulation step and described in WorkBookSimulation.

Previous Releases

- 13.0.30
- 12.0.6
- 11.0.5
- 10.0.1

Complete: 🟩
Responsible: Steve Lloyd
Author: James Catmore 03 Feb 2005
Contributors: James Catmore, Chris Collins Tooth, Gernot Krobath, Steve Lloyd
Last significantly modified by: Gernot Krobath 30 Aug 2006
Last reviewed by: Steve Lloyd 16 Oct 2008

3.5 Reconstructing Events with Athena

This section describes how to reconstruct previously digitized (or real) events with Athena. Reconstruction is the process whereby the raw data **Digits**, such as times and voltages, are reconstructed into tracks and energy deposits as Event Summary Data (ESD).

You first need to set up your account as described in WorkBookSetAccount and check out a package to work in such as **UserAnalysis** as described in WorkBookRunAthenaHelloWorld. The digitization of events is described in WorkBookDigitization. For further information see RunningReconstruction.

3.5.1 Running Reconstruction in Athena

Go to your run directory such as:

```
> cd testarea/14.2.23
> cd PhysicsAnalysis/AnalysisCommon/UserAnalysis/run
```

(but note that in principle reconstruction can be run in any directory)

If you intend to do a lot of reconstruction you might find it useful to run a script to copy/create a few necessary files in your run directory:

```
RecExCommon_links.sh
```

Create a small job options file such as `myRecOptions.py` with the following lines in it, with the input and output files set appropriately:

```
###############################################################
# Detector description
 DetDescrVersion='ATLAS-CSC-02-02-00'
 OutputLevel=ERROR

doCBNT=False

#number of Event to process (-1 is all)
EvtMax = 10
SkipEvents = 0

# the input data file
PoolRDOInput = [ "g4digi.pool.root" ]

# The ESD output file name
PoolESDOutput = "esd.pool.root"

 include ("RecExCommon/RecExCommon_topOptions.py")
###############################################################
```

The line `DetDescrVersion`'ATLAS-CSC-02-02-00'= defines the version of the Detector Description to use as described in AtlasGeomDBTags#ATLAS_Geom_DB_Tag_contents.

You can now run athena:

```
> athena.py myRecOptions.py > athena_rec.out
```

You should have a file of ESD (this was 10 Z0→e+e- events):

```
-rw-r--r--  1 lloyd lloyd   9273382 May 23 20:14 esd.pool.root
-rw-r--r--  1 lloyd lloyd  19172652 May 23 20:05 g4digi.pool.root
-rw-r--r--  1 lloyd lloyd  12996310 May 23 20:02 g4hits.pool.root
-rw-r--r--  1 lloyd lloyd    254185 May 23 16:48 pythia.pool.root
```

(Instead of creating myRecOptions.py, another equivalent procedure is to use file myTopOptions.py and adapt it to your needs, and run this (this is the default file))

You can produce Analysis Object Data (AOD) at the same time as producing ESD if in your `myRecOptions.py` file you change

```
doWriteAOD = True
```

and uncomment

```
PoolAODOutput = "aod.pool.root"
```

Full details of how ESD, AOD and TAG production in Athena can be found in UserAnalysisTest.

3.5.2 Produce ESD and Athena Aware NTuple in the same job

You can also produce an Athena Aware NTuple in the same job by adding the following to your `myRecOptions.py`.

```
# specify CBNT AA is requested
CBNTAthenaAware=True
# name of root output file
RootNtupleOutput="AANTuple.root"
```

Then rerun athena and you should have ESD and AANT:

```
$ ls -l *.root
-rw-r--r--    1 tmaeno   zp          14412 Feb 16 17:11 AANTuple.root
-rw-r--r--    1 tmaeno   zp        3604756 Feb 16 17:11 ESD.pool.root
-rw-r--r--    1 tmaeno   zp            382 Feb 16 17:11 histo.root
-rw-r--r--    1 tmaeno   zp         171369 Feb 16 17:11 ntuple.root
```

Previous Releases

- 13.0.30
- 12.0.6
- 11.0.5
- 10.0.1

Complete: 🟩
Responsible: Steve Lloyd
Author: Steve Lloyd 27 Apr 2005
Contributors: Gernot Krobath, Steve Lloyd, David Rousseau
Last significantly modified by: Steve Lloyd 08 Feb 2008
Last reviewed by: Steve Lloyd 16 Oct 2008

3.6 Producing AOD with Athena

This section describes how to produce Analysis Object Data (AOD) with Athena. AOD is a summary of the reconstructed ESD.

Full details of ESD, AOD and TAG production in Athena can be found in AtlasProtected.UserAnalysisTest.

Note: There may be problems producing 13.0.30 AOD from 13.0.30 ESD. You can however produce AOD during reconstruction and this seems to be OK.

You first need to set up your account as described in WorkBookSetAccount and check out a package to work in such as **UserAnalysis** as described in WorkBookRunAthenaHelloWorld. The reconstruction of events is described in WorkBookReconstruction.

Go to your run directory such as:

```
> cd testarea/14.2.23
> cd PhysicsAnalysis/AnalysisCommon/UserAnalysis/run
```

You need to create a file such as `myAODOptions.py` with appropriate input and output files:

```
###############################################
 DetDescrVersion='ATLAS-CSC-02-02-00'

readESD = True
doWriteESD=False
doWriteAOD = True
doWriteTAG=False
doAOD = True


EvtMax = -1

PoolESDInput = [ "esd.pool.root" ]
PoolAODOutput = "aod.pool.root"

include ("RecExCommon/RecExCommon_topOptions.py")
##################################################
```

The line `DetDescrVersion`'ATLAS-CSC-02-02-00'= defines the version of the Detector Description to use as described in AtlasGeomDBTags#ATLAS_Geom_DB_Tag_contents.

You can now run athena:

```
> athena.py myAODOptions.py  > athena_aod.out
```

You should have a file of AOD (this was 10 Z0→e+e- events):

```
-rw-r--r--  1 lloyd lloyd   2183312 May 28 15:32 aod.pool.root
-rw-r--r--  1 lloyd lloyd   9273382 May 23 20:14 esd.pool.root
-rw-r--r--  1 lloyd lloyd  19172652 May 23 20:05 g4digi.pool.root
-rw-r--r--  1 lloyd lloyd  12996310 May 23 20:02 g4hits.pool.root
-rw-r--r--  1 lloyd lloyd    254185 May 23 16:48 pythia.pool.root
```

In order to check that everything is alright you can run a standard analysis on the AOD as described in WorkBookReadingAOD.

Previous Releases

- 13.0.30
- 12.0.6
- 11.0.5
- 10.0.1

Complete: 🟩
Responsible: Steve Lloyd
Author: Steve Lloyd 19 May 2005
Contributors: Steve Lloyd
Last significantly modified by: Steve Lloyd 20 Jan 2006
Last reviewed by: Steve Lloyd 16 Oct 2008

3.7 Producing Fast Simulation AOD with Athena

This section describes how to produce Analysis Object Data (AOD) with Athena using the Fast Simulation **Atlfast** directly from the generated events.

You first need to set up your account as described in WorkBookSetAccount and check out a package to work in such as **UserAnalysis** as described in WorkBookRunAthenaHelloWorld. The generation of events is described in WorkBookGeneration.

3.7.1 Running Atlfast in Athena

The instructions below are to produce the fast AOD in standalone mode. Furthermore, the fast Track-Particles are not saved in the fast AOD by default. But there is an AOD flag, to add them if necessary. See UserAnalysisTest for further details. Note from the release 11.1.0, the fast and full AOD are saved by default in the same output file. The Atlfast home page has some useful information. Most noteably on the page Running Atlfast.

Go to your run directory such as:

```
> cd testarea/14.2.23
> cd PhysicsAnalysis/AnalysisCommon/UserAnalysis/run
```

You need to obtain the following file via:

```
> get_files POOLtoAtlfasttoAOD.py
> get_files Atlfast_ReadPOOLFile.py
```

Change the input file appropriately in `Atlfast_ReadPOOLFile.py`, e.g.

```
EventSelector.InputCollection = [ "pythia.pool.root" ]
```

Replace the following in `POOLtoAtlfasttoAOD.py`:

```
#include ( "AtlfastAlgs/Atlfast_ReadPOOLFile.py" )
```

by

```
include ( "./Atlfast_ReadPOOLFile.py" )
```

You can now run athena:

```
> athena.py  POOLtoAtlfasttoAOD.py > athena_fast.out
```

You should have a file of AOD (this was 10 Z0→e+e- events):

```
-rw-r--r--  1 lloyd hep  172428 Jan 19 16:23 AOD.pool.root
-rw-r--r--  1 lloyd hep  358735 Jan 19 13:49 pythia.pool.root
```

(Not sure what you do with this as AnalysisSkeleton gives all empty histograms).

3.7.2 Producing Fast Simulation CBNT with Athena

Using the `POOLtoAtlfasttoAAN.py` script in place of `POOLtoAtlfasttoAOD.py` in the above instructions produces a CBNTAA (Athena-Aware ROOT N-tuple).

The produced CBNT file can be read by root without any further processing

Previous Releases

- 13.0.30
- 12.0.6
- 11.0.5
- 10.0.1

Complete: 🟩
Responsible: Steve Lloyd
Author: Nathan Triplett 22 Jun 2005
Contributors: Andrew Hamilton, Steve Lloyd, Nathan Triplett
Last significantly modified by: Andrew Hamilton 24 Sep 2007
Not yet reviewed

3.8 Reading AOD with Athena

This section describes how to read Analysis Object Data (AOD) with Athena. AOD is a summary of the reconstructed ESD.

You first need to set up your account as described in WorkBookSetAccount and check out a package to work in such as **AtlasProtected.UserAnalysis** as described in WorkBookRunAthenaHelloWorld. The production of AOD is described in WorkBookProducingAOD. This uses a template analysis **AnalysisSkeleton** that you can customise later for your own purposes. For much more information on read AOD and doing analysis see the Physics Analysis Workbook. The AOD classes are described in detail in AODClassSummary.

Obtain the following file:

```
> get_files AnalysisSkeleton_topOptions.py
```

Change the following lines to the appropriate file name:

```
EventSelector.InputCollections = [
                                  "aod.pool.root"
                                 ]
```

Now run athena:

```
> athena.py AnalysisSkeleton_topOptions.py > athena_analysis.out
```

You should have a file `AnalysisSkeleton.aan.root` which you can inspect via Root

Note: In Release 13.0.30 the electron container name changed from `ElectronCollection` to `ElectronAODCollection`.

Note: If you are running at a site other than CERN, and if the AOD file is in a AFS directory at CERN, e.g., /afs/cern.ch/atlas/maxidisk/d129/WorkBookExample, you don't need to copy the file to your local disk. Instead, do the following:

- klog username@cern.ch (and give your CERN password) - this will let you read files over AFS. If you have problems, check with your local adminstrator.
- Use the full path of the filename, i.e., `EventSelector.InputCollections = [/afs/cern.ch/atlas/maxidisk/d`

Note: In release 14.2.23 (and possibly others) the electron histograms and ntuples are not filled by AnalysisSkeleton because the call to `electronSkeleton()` is commented out. To overcome this checkout a later version of UserAnalysis (or uncomment the line in the current one):

```
cmt co -r UserAnalysis-00-13-10 PhysicsAnalysis/AnalysisCommon/UserAnalysis
cd PhysicsAnalysis/AnalysisCommon/UserAnalysis/cmt
source setup.sh
gmake
```

3.8.1 Using ROOT

For a comprehensive introduction to ROOT see the ROOT Basic Tutorial.

```
> root
root [0] b = new TBrowser();
```

- Click on the run folder on the left (2nd down)
- Double click on AnalysisSkeleton.aan.root
- Click on ROOT files
- Double click on AnalysisSkeleton.aan.root
- Double click on Electron
- Double click on each histogram in turn

or

```
> root AnalysisSkeleton.aan.root
root [0] b = new TBrowser();
```

- Click on ROOT files (on left panel)
- Double click on AnalysisSkeleton.aan.root
- Double click on Electron
- Double click on each histogram in turn

To quit:

```
root [1] .q
```

The analysis algorithm itself is described in more detail in the AtlasProtected.PhysicsAnalysisWorkBook.

3.8.2 More AOD Samples

Files containing 100 Z→ee AOD events are available at CERN at:

```
/afs/cern.ch/user/l/lloyd/public/zee100_14.2.23_aod.pool.root
/afs/cern.ch/user/l/lloyd/public/zee100_13.0.30_aod.pool.root
/afs/cern.ch/user/l/lloyd/public/zee100_12.0.5_aod.pool.root
```

You can either read these directly on lxplus or copy them to your own space/institute. These events are not part of the official production and should only be used for testing.

You may need to insert the name into the file catalogue:

```
pool_insertFileToCatalog /afs/cern.ch/user/l/lloyd/public/zee100_14.2.23_aod.
    pool.root
```

Previous Releases

- 13.0.30
- 12.0.6

– Main.SteveLloyd - 26 July 2005

Complete: 🟩
Responsible: Steve Lloyd
Author: Ketevi Assamagan 21 Feb 2005
Contributors: Ketevi Assamagan, Steve Lloyd
Last significantly modified by: Steve Lloyd 08 May 2007
Last reviewed by: Steve Lloyd 16 Oct 2008

Chapter 4

Using the Grid

4.1 Starting on the Grid

Before you can use the Grid you need to obtain a **Digital Certificate** and then join the ATLAS Virtual Organisation (VO).

4.1.1 Getting a Digital Certificate

In order to use the Grid you need a **Digital Certificate** (sometimes called a PKI or X509 certificate) that acts as a passport and says who you are (know as **Authentication**). You obtain a Digital Certificate from your National **Certification Authority** (CA). Usually the procedure is the following but you should always follow instructions for your particular CA:

- Go to one of the links below and find your National CA:
  - LCGG, NorduGrid and any country that belongs to IGTF: `http://lcg.web.cern.ch/LCG/digital.htm`
  - US-GRID (United States): https://www.racf.bnl.gov/docs/howto/grid/
- Retrieve a certificate from the site and request a (User) certificate
- Go to your local RA (Registration Authority) with your real passport or similar ID to authenticate you.
- Once authenticated you should get a reply by email
- Following the link in the mail, retrieve the certificate (don't worry if you are using Mozilla/Firefox and you get no answer).
- Find the certificate in your browser. This is very browser dependent. It will be something like *Edit* or *Tools* → *Preferences* or *(Internet) Options* → *Advanced* → *Security* or *Encryption* → *View Certificates* → *Your Certificates*. From here you will be able to 'backup' your certificate to export it to other browsers or to protect it with a password or (see passphrase, passphrase1) . You can find some information about loading certificates with different browsers at http://grid-it.cnaf.infn.it/fileadmin/users/certmgr/certmgr.html.

Note:

- It may take a few days to obtain a certificate.
- Always use the **same browser** on the **same computer** during these operations.

4.1.2 Joining the ATLAS Virtual Organisation

Once you have a certificate you need to join a **Virtual Organisation** (VO) - in this case **atlas**. Being a member of a VO is like having Visas in your passport that say what you can do (known as **Authorisation**).

ATLAS uses 3 different Grid implementations - OSG in the US, NorduGrid and the LCGG. To join the LCGG you need to

go to https://lcg-voms.cern.ch:8443/vo/atlas/vomrs. New members should use the Phase 1 registration link. Existing members who want to renew a certificate etc should click on the [+] next to Member Info and then "Re-sign Grid and VO AUPs". To join OSG follow the how to join the ATLAS VO procedure.

When asked, select only the following groups and NO ROLES (unless you really know what you're asking for):

- /atlas
- /atlas/lcg1

These groups are enough for full data access and job handling within the ATLAS VO as a normal user. US ATLAS users should also select the additional group:

- /atlas/usatlas

4.1.3 Preparing to use the LCGG or US-Grid

Once you have your Digital Certificate (probably inside your Web Browser) you need to export it. How you do this will depend on your Web Browser as described above. You should end up with something like `mycert.pfx`. In order to use the Grid you need access to a local **User Interface** (UI) machine that has the LCGG Middleware installed. Copy your certificate to this UI machine and under your home directory on the UI create a `.globus` directory.

User Interface

You can use lxplus (or another machine under AFS) as a Grid User Interface (UI) if you issue the command:

```
> source /afs/cern.ch/project/gd/LCG-share/current/etc/profile.d/grid_env.sh  [
   or .csh]
```

Note: Because of problems with versions of SL3/SL4 you may have to substitute "previous" for "current". (or vice versa!) (Also don't use the option brace_ccl in your zsh. The gridenv functions don't work then, the X509_USER_PROXY and JAVA_PATH variable will not be set)

Instructions for installing a UI can be found at http://glite.web.cern.ch/glite/packages/R3.1/deployment/glite-UI/glite-UI.asp. There is also a tarball installation that does not require root access at https://twiki.cern.ch/twiki/bin/view/LCG/TarUIInstall, suitable for installation on a laptop for instance.

See also GridUserInterfaces.

Preparing your Certificate

You need to convert your certificate into the correct form using:

```
> openssl pkcs12 -in mycert.pfx -clcerts -nokeys -out usercert.pem
> openssl pkcs12 -in mycert.pfx -nocerts -out userkey.pem
> chmod 400 userkey.pem
> chmod 444 usercert.pem
```

then move these two files to the .globus directory (If you haven't got one then mkdir ~/.globus). You probably need to remember two passwords, one for the original certificate and one for the converted one. If all is well try:

```
> voms-proxy-init
```

(this used to be grid-proxy-init) which should give something like this:

```
Your identity: /C=UK/O=eScience/OU=QueenMaryLondon/L=Physics/CN=steve lloyd
Enter GRID pass phrase for this identity:
Creating proxy ...............................................
    Done
Your proxy is valid until: Tue Apr 12 22:03:18 2005
```

You need to do this once every session or after 12 hours have elapsed. You might get an error (that appears to be harmless):

```
Cannot find file or dir: /afs/cern.ch/user/l/lloyd/.glite/vomses
```

4.1.4 Preparing to use the NorduGrid

Obtaining NorduGrid/ARC client tools

In order to submit jobs, copy files and do other useful things on NorduGrid and other !ARC-enabled resources, you need an appropriate set of client tools. You can download them from the NorduGrid downloads site: http://ftp.nordugrid.org/download.

Click **Quick start: standalone client** button and scroll to the bottom of the page. From the table offering different Linux flavors, click the one of your liking; **for SLC3, select redhat-3WS**. Save the file (tar.gz), or unpack it directly to any place you prefer. You will need 15 to 21 MBB of disk space, depending on the version.

After unpacking, go down the newly created directory, e.g. nordugrid-standalone-0.4.5, make the setup script an executable chmod u+x setup.[c]sh, and then run

```
> source setup.[c]sh
```

You will have to run this for every new opened shell session, so it makes sense to add this line to your .ba—cshrc initialisation file.

Don't forget to run

```
> grid-proxy-init
```

to create your proxy or to renew it. The default Grid proxy lifetime is 12 hours.

Originally taken from https://lcg-registrar.cern.ch/pki_certificates.html

Complete: ▰▰▰▰▱

Responsible: Steve Lloyd
Author: Steve Lloyd 03 Feb 2005
Contributors: Steve Lloyd, Oxana Smirnova
Last significantly modified by: Steve Lloyd 19 Apr 2007
Not yet reviewed

4.2 Running a Grid Job

4.2.1 Running a job on the LCGG Grid

You need access to a computer that is runing the User Interface (UI) middleware as described in WorkBookStartingGrid#UserInterface.

First create a file which contains the **Job Description Language** commands to run the job, say `helloworld.jdl`:

```
#############Hello World#################
Executable = "/bin/echo";
Arguments = "Hello welcome to the Grid ";
StdOutput = "hello.out";
StdError = "hello.err";
OutputSandbox = {"hello.out","hello.err"};
VirtualOrganisation = "atlas";
########################################
```

The **Executable** is the execuatable you want to run (normally a script but in this case just a simple command), **Arguments** are any arguments you wish to give to the comman/script. The **InputSandbox** is a list of files that you want to send with the job and the **OutputSandbox** is the list of files you want to retrieve when it has finished.

Now make a temporary version of your Grid Certificate (you need to do this once every session or after 12 hours have elapsed):

```
> voms-proxy-init --voms atlas
Enter GRID pass phrase:
Your identity: /C=UK/O=eScience/OU=QueenMaryLondon/L=Physics/CN=steve lloyd
Creating temporary proxy ......................................... Done
Contacting  voms.cern.ch:15001 [/DC=ch/DC=cern/OU=computers/CN=voms.cern.ch] "
    atlas" Done
Creating proxy ............................... Done
Your proxy is valid until Wed Oct  8 03:58:11 2008
```

You might get an error (that appears to be harmless):

```
Cannot find file or dir: /afs/cern.ch/user/l/lloyd/.glite/vomses
```

Now submit the job:

```
> glite-wms-job-submit -a -o jobIDfile helloworld.jdl
======================= glite-wms-job-submit Success =======================

The job has been successfully submitted to the WMProxy
Your job identifier is:

https://lb106.cern.ch:9000/_qXQgXqlXVNNChT5A-M22g
```

```
The job identifier has been saved in the following file:
/afs/cern.ch/user/l/lloyd/jobIDfile

==========================================================================
```

The `jobIDfile` is a file that contains the job identifiers so that you don't need to ever type them in. You probably want to delete this file from time to time.

You can find out the status by typing:

```
>  glite-wms-job-status -i jobIDfile

------------------------------------------------------------------
1 : https://lb106.cern.ch:9000/2kC77gJL6aze8n8f1fV7Rw
2 : https://lb106.cern.ch:9000/_qXQgXqlXVNNChT5A-M22g
a : all
q : quit
------------------------------------------------------------------

Choose one or more jobId(s) in the list - [1-2]all:2

*************************************************************
BOOKKEEPING INFORMATION:

Status info for the Job : https://lb106.cern.ch:9000/_qXQgXqlXVNNChT5A-M22g
Current Status:     Done (Success)
Logged Reason(s):
    -
    - Job terminated successfully
Exit code:          0
Status Reason:      Job terminated successfully
Destination:        ce-3-fzk.gridka.de:2119/jobmanager-pbspro-atlasXS
Submitted:          Tue Oct  7 15:58:15 2008 CEST
*************************************************************
```

Once the job has terminated you can retrieve the output (jo is the directory to save the job output):

```
> glite-wms-job-output -i  jobIDfile --dir jo
------------------------------------------------------------------
1 : https://lb106.cern.ch:9000/2kC77gJL6aze8n8f1fV7Rw
2 : https://lb106.cern.ch:9000/_qXQgXqlXVNNChT5A-M22g
a : all
q : quit
------------------------------------------------------------------

Choose one or more jobId(s) in the list - [1-2]all (use , as separator or - for
   a range): 2

Connecting to the service https://wms105.cern.ch:7443/glite_wms_wmproxy_server

================================================================================

                        JOB GET OUTPUT OUTCOME

Output sandbox files for the job:
```

```
https://lb106.cern.ch:9000/_qXQgXqlXVNNChT5A-M22g
have been successfully retrieved and stored in the directory:
/afs/cern.ch/user/l/lloyd/jo
```

and inspect the results:

```
> ls -lt jo
total 1
-rw-r--r--   1 lloyd zp 32 Oct  7 16:05 hello.out
-rw-r--r--   1 lloyd zp  0 Oct  7 16:05 hello.err

> more jo/hello.out
Hello welcome to the Grid
```

Note: Because of incompatible versions of python, if you source the **dq2** setup scripts then the `edg` or `glite` commands will not work. You should use dq2 in a different terminal window than the one you use for submitting the jobs. This problem does not occur if you use Ganga for job submission.

4.2.2 EDG versus gLite commands

The original EDG (European Data Grid) middleware used commands like `edg-job-xxx` which communicated with an RB (Resource Broker). These have been replaced with gLite commands that communicate with a WMS (Workload Management System). The correspondence between the two sets of commands is shown below:

gLite Command	EDG Command
glite-wms-job-submit -a -o jobIDfile helloworld.jdl	edg-job-submit –vo atlas -o jobIDfile helloworld.jdl
glite-wms-job-status -i jobIDfile	edg-job-status -i jobIDfile
glite-wms-job-output -i jobIDfile–dir jo	edg-job-get-output -dir . -i jobIDfile
glite-wms-job-cancel -i jobIDfile	edg-job-cancel -i jobIDfile
glite-wms-job-list-match -a helloworld.jdl	edg-job-list-match helloworld.jdl

Complete: ▨
Responsible: Steve Lloyd
Author: Steve Lloyd 24 Jan 2006
Contributors: Steve Lloyd
Last significantly modified by: Steve Lloyd 07 Oct 2008
Not yet reviewed

4.3 Running Athena on the Grid

NOTE: **The official way of running Athena analysis jobs on the Grid** is to use GANGA and/or pAthena. Please see instructions in the AtlasProtectedPhysicsAnalysisWorkBook.

Note: Because of incompatible versions of python, if you source the **dq2** setup scripts then the `edg` or `glite` commands will not work. You should use dq2 in a different terminal window than the one you use for submitting the jobs. This problem does not occur if you use GANGA for job submission.

4.3.1 Running HelloWorld on the LCGG Grid

You need access to a computer that is runing the User Interface (UI) middleware as described in WorkBookStartingGrid#UserInterface.

Creating the Necessary Files

First obtain the HelloWorldOptions.py Job Options file, either from the Release or Kit as described in WorkBookRunAthenaHelloWorld or directly from the Web interface to the CVS repository: http://cern.ch/atlas-computing/links/buildDirectory and navigating to (substituting version numbers where appropriate):
AtlasCore → 14.2.23 → Control → AthenaExamples → AthExHelloWorld → share

You then need to create two files, the first is a script which you will run, say hello.sh, containing:

```
#!/bin/bash
# Script to run AthenaHelloWorld on the Grid

source $VO_ATLAS_SW_DIR/software/14.2.23/setup.sh
source $SITEROOT/AtlasOffline/14.2.23/AtlasOfflineRunTime/cmt/setup.sh

athena.py HelloWorldOptions.py
```

and a second which contains the **Job Description Language** commands to run the job, say hello.jdl:

```
############# Athena #################
Executable = "hello.sh";
StdOutput = "hello.out";
StdError = "hello.err";
InputSandbox = {"hello.sh","HelloWorldOptions.py"};
OutputSandbox = {"hello.out","hello.err", "CLIDDBout.txt"};
Requirements = Member("VO-atlas-offline-14.2.23-i686-slc4-gcc34-opt", other.
    GlueHostApplicationSoftwareRunTimeEnvironment);
#####################################################
```

Note: that there has been a subtle change in the way to specify the release you need in the 'Requirements' line: -atlas-offline-14.2.23-i686-slc4-gcc34-opt used to be VO-atlas- **production** -14.2.23 - 13.0.30, VO-atlas- **offline** -12.0.2 and VO-atlas- **release** -11.0.5.

The **InputSandbox** is a list of files that you want to send with the job and the **OutputSandbox** is the list of files you want to retrieve when it has finished. The **Requirements** ensures that the job is submitted to a site that has the required version of the ATLAS software installed.

Submitting the Job

Now make a temporary version of your Grid Certificate (you need to do this once every session or after 12 hours have elapsed):

```
> voms-proxy-init
Enter GRID pass phrase:
Your identity: /C=UK/O=eScience/OU=QueenMaryLondon/L=Physics/CN=steve lloyd
Creating temporary proxy ........................................ Done
Contacting  voms.cern.ch:15001 [/DC=ch/DC=cern/OU=computers/CN=voms.cern.ch] "
    atlas" Done
Creating proxy .............................. Done
Your proxy is valid until Wed Oct  8 03:58:11 2008
```

Now submit the job:

```
> edg-job-submit --vo atlas -o jobIDfile hello.jdl
> glite-wms-job-submit -a -o jobIDfile helloworld.jdl
====================== glite-wms-job-submit Success ======================

The job has been successfully submitted to the WMProxy
Your job identifier is:

https://lb106.cern.ch:9000/_qXQgXqlXVNNChT5A-M22g

The job identifier has been saved in the following file:
/afs/cern.ch/user/l/lloyd/jobIDfile

==========================================================================
```

The `jobIDfile` is a file that contains the job identifiers so that you don't need to ever type them in. You probably want to delete this file from time to time.

Checking the Status of the Job

You can find out the status by typing:

```
>  glite-wms-job-status -i jobIDfile

------------------------------------------------------------------

1 : https://lb106.cern.ch:9000/2kC77gJL6aze8n8f1fV7Rw
2 : https://lb106.cern.ch:9000/_qXQgXqlXVNNChT5A-M22g
a : all
q : quit
------------------------------------------------------------------

Choose one or more jobId(s) in the list - [1-2]all:2

*************************************************************
BOOKKEEPING INFORMATION:

Status info for the Job : https://lb106.cern.ch:9000/_qXQgXqlXVNNChT5A-M22g
Current Status:     Done (Success)
Logged Reason(s):
    -
    - Job terminated successfully
Exit code:          0
Status Reason:      Job terminated successfully
Destination:        ce-3-fzk.gridka.de:2119/jobmanager-pbspro-atlasXS
Submitted:          Tue Oct  7 15:58:15 2008 CEST
*************************************************************
```

Retreiving the Output

Once the job has terminated you can retrieve the output and inspect the results:

```
> glite-wms-job-output -i  jobIDfile --dir jo
------------------------------------------------------------------

1 : https://lb106.cern.ch:9000/2kC77gJL6aze8n8f1fV7Rw
```

```
2 : https://lb106.cern.ch:9000/_qXQgXqlXVNNChT5A-M22g
a : all
q : quit
------------------------------------------------------------------------------

Choose one or more jobId(s) in the list - [1-2]all (use , as separator or - for
    a range): 2

Connecting to the service https://wms105.cern.ch:7443/glite_wms_wmproxy_server

==============================================================================

                       JOB GET OUTPUT OUTCOME

Output sandbox files for the job:
https://lb106.cern.ch:9000/_qXQgXqlXVNNChT5A-M22g
have been successfully retrieved and stored in the directory:
/afs/cern.ch/user/l/lloyd/jo
> ls -lt jo
total 1
-rw-r--r--  1 lloyd zp 32 Oct  7 16:05 hello.out
-rw-r--r--  1 lloyd zp  0 Oct  7 16:05 hello.err
```

Other Useful Commands

Other useful commands are

```
> glite-wms-job-cancel -i jobIDfile
```

and

```
> glite-wms-job-list-match -a  athena.jdl
```

which tells you which (if any) sites match your requirements without actually submitting the job.

ATLAS Versions

To find out what versions of ATLAS software are installed at each site you can type:

```
lcg-infosites --vo atlas tag
```

or visit https://atlas-install.roma1.infn.it/atlas_install/

Temporary Space

If your job needs temporary space please do **not** use /tmp. There should be a special area set up as $EDG_WL_SCRATCH (LCG) or $OSG_WN_TMP (OSG) that you can use for small amounts of temporary data. If there is a lot of data or you need to keep it you should store it on a Storage Element (SE) as described in WorkBookFullChainGrid.

Grid Error Messages

Some of Grid error messages and their causes/cures can be found in GridErrorMessages.

4.3.2 Running HelloWorld on the NorduGrid

This assumes you have a valid Grid certificate and are a member of ATLASS VOO or a NorduGrid VOO. You need to obtain the NorduGrid/ARC client tools as described in WorkBookStartingGrid#NorduGridSettingUp.

Preparing HelloWorld job description

The most flexible way of submitting jobs, as with any batch system, is to prepare your shell script that in turn calls athena. Here is an example of such script for HelloWorld job:

```
get_files HelloWorldOptions.py
athena.py -s HelloWorldOptions.py
```

Store these lines in a file by any name, say, HelloAthena.sh

Submitting HelloWorld job

ATLAS Software is pre-installed at many NorduGrid sites and is configured automatically, as soon as you request it in your instructions. Job submission instructions must contain the name of the executable file (HelloAthena.sh in our case), its original location (current directory, in this example), requested software, and name of the standard output file, if you wish to see the results. Many other criteria can be specified, but it is beyond the scope of this example.

The job submisson is done via **ngsub** command:

```
> ngsub -t 5 '&(executable="HelloAthena.sh")(runtimeenvironment>"APPS/HEP/ATLAS
   -10.0.1")(inputfiles=("HelloAthena.sh" ""))(stdout="Hello.out")'
```

If you use nordugrid-arc-standalone version 0.5.14 and above, add option `-e` in front of the quoted string.

Type `ngsub -v` to get version number

If you don't want to type all the job description string in the quotes on the command line, save it in a file. Typically, such files have extension `xrsl`, but it can be anything else. If your job description is saved in a file called like HelloAthena.xrsl, do:

```
> ngsub -t 5 -f HelloAthena.xrsl
```

If you use nordugrid-standalone version 0.5.14 and above, do not type `-f` in front of the file name.

If your executable script HelloAthena.sh is not in your current directory, add the relative or an absolute path before its name in `inputfiles` instruction. If you wish to use another ATLASS software version, change the value in the `runtimeenvironment` string accordingly.

Successful submision will print out the job handle (`jobid`):

```
Job submitted with jobid gsiftp://grid00.unige.ch:2811/jobs
   /785711305136841463641651
```

Managing the job

To manage the job (watch, kill, retrieve, resubmit etc), always use the `jobid` as above.

To watch the status of the job, use **ngstat**:

```
> ngstat gsiftp://grid00.unige.ch:2811/jobs/785711305136841463641651

Job gsiftp://grid00.unige.ch:2811/jobs/785711305136841463641651
  Status: FINISHED
```

To check the otput of the job while it runs, or without retrieving it, use **ngcat**:

```
> ngcat gsiftp://grid00.unige.ch:2811/jobs/785711305136841463641651

...
AthenaEventLoopMgr   INFO   ===>>> end of event 9   <<<===
 -+-  10 A theApp.exit( result.isFailure() )
HistorySvc           INFO Service finalised successfully
HelloWorld           INFO finalize()
EventSelector        INFO finalize
ToolSvc              INFO Removing all tools created by ToolSvc
ApplicationMgr       INFO Application Manager Finalized successfully
ApplicationMgr       INFO Application Manager Terminated successfully
```

If job status is `FINISHED`, you can retrieve the result using **ngget**:

```
> ngget gsiftp://grid00.unige.ch:2811/jobs/785711305136841463641651

ngget: downloading files to /home/janedoe/mygridstuff/785711305136841463641651
ngget: download successful - deleting job from gatekeeper.
```

For more information, see `man ngsub`, and visit the NorduGrid site for documentation, such as the User Guide.

Advanced example: using your own joboptions

You can upload your own files to the job. If you have a modified `MyHelloWorldOptions.py`, you should add it to the `inputfiles` list in job description, and modify the executable script `HelloAthena.sh` accordingly. Let's create a new script `athena.sh`, generalized to use different !jobOptions:

```
athena.py -s $1
```

It is certainly more convenient to keep job descriptions in a file. An example job description file `hello1.xrsl` is:

```
&
(executable="athena.sh")
(arguments="MyHelloWorldOptions.py")
(stdout="HelloWorld.out")
(inputfiles=
  ("athena.sh" "")
  ("MyHelloWorldOptions.py" "")
)
(runTimeEnvironment="APPS/HEP/ATLAS-10.4.0")
(jobname="My Hello World job")
```

Use `ngsub`, as above, to submit this job.

Previous Releases

- 13.0.30
- 12.0.6
- 11.0.5

Complete: ▨
Responsible: Steve Lloyd
Author: Steve Lloyd 03 Feb 2005
Contributors: Steve Lloyd, Oxana Smirnova
Last significantly modified by: Steve Lloyd 24 Jan 2006
Not yet reviewed

4.4 Running the Full Chain on the Grid

NOTE: **The official way of running Athena analysis jobs on the Grid** is to use GANGA and/or pAthena. Please see instructions in the AtlasProtectedPhysicsAnalysisWorkBook.

To run the full chain on the Grid you proceed as described in WorkBookAthenaGrid substituting suitable job options files where necessary. i.e. to generate events a suitable script called athena_gen.sh might be:

```
#!/bin/bash
source $VO_ATLAS_SW_DIR/software/14.2.23/setup.sh
source $SITEROOT/AtlasOffline/14.2.23/AtlasOfflineRunTime/cmt/setup.sh
cp $SITEROOT/AtlasOffline/14.2.23/InstallArea/share/PDGTABLE.MeV .
# Run the job:
athena.py jobOptions_gen.py
```

and a suitable athena_gen.jdl file:

```
############ Athena #################
Executable = "athena_gen.sh";
StdOutput = "athena_gen.out";
StdError = "athena_gen.err";
InputSandbox = {"athena_gen.sh","jobOptions_gen.py"};
OutputSandbox = {"athena_gen.out", "athena_gen.err", "pythia.pool.root", "
    CLIDDBout.txt"};
Requirements = Member("VO-atlas-production-14.2.23",
 other.GlueHostApplicationSoftwareRunTimeEnvironment);
####################################################
```

However, this will ship the output events `pythia.pool.root` back to you in the output sandbox and if the file is large, as it will be after simulation, you may not want to store it locally. In which case you should copy it to a Grid Storage Element (SE) and register it for future use. You can get a list of SEs using the `lcg-infosites` command:

```
> lcg-infosites --vo atlas se
**************************************************************
These are the related data for atlas: (in terms of SE)
**************************************************************

Avail Space(Kb) Used Space(Kb)  Type    SEs
----------------------------------------------------------
...
3638435648      2217730240      disk    lcgce.ijs.si
1447509920      12727816        disk    seitep.itep.ru
...
```

Remove pythia.pool.root from the OutputSandbox of the jdl and add a line to the end of your athena_gen.sh to use the lcg-cr (copy and register) command:

```
# Run the job:
athena.py jobOptions_gen.py
# Define logical file catalog and logical file namespace
export LFC_HOST=lfc0448.gridpp.rl.ac.uk
lfc-mkdir -p /grid/atlas/users/$USERNAME
```

where $USERNAME is your login name (or preferred identifier). This follows the evolution of the LCG middleware: lfns have now a mandatory directory structure and the logical file catalog is not unique anymore, but distributed across the Grid. You can extend the namespace by adding more directories. LFC_HOST at CERN is lfc-atlas.cern.ch

To save the generated events do:

```
lcg-cr -v --vo atlas -d dcache.gridpp.rl.ac.uk
   -l lfn:/grid/atlas/users/$USERNAME/lloyd_pythia.pool.root file://`pwd`/pythia
       .pool.root
```

where the -d option specifies the name of the SE and -l any logical file name you want to give it.

Note the ‘pwd‘ (not ’pwd’) substitutes the current path into the file name.

In the next step of the chain you can read these events by including the following in a second shell script:

```
...
# Get the input
lcg-cp -v --vo atlas lfn:lloyd_pythia.pool.root file://`pwd`/pythia.pool.root
...
# Insert the file into the FileCatalog
pool_insertFileToCatalog pythia.pool.root
# Run the job:
athena.py jobOptions_xxx.py
# Save the output events
lcg-cr -v --vo atlas -d dcache.gridpp.rl.ac.uk
  -l lfn:/grid/atlas/users/$USERNAME/lloyd_fast_aod.pool.root file://`pwd`/
      fast_aod.pool.root
...
```

where the second Job Options file (say) `jobOptions_xxx.py` contains the correct name of the input `pythia.pool.root`.

Note: In Release 13.0.30 you should do the `lcg-cp` **before** sourcing the ATLAS setup scripts otherwise `lcg-cp` may seg fault.

After this you should have 2 files registered. You can find where they are by using Don Quijote:

```
> /afs/cern.ch/atlas/offline/external/DQClient/dms3/dms3.py search '*lloyd*'
Using home grid:  ['http://atlfarm009.mi.infn.it:11122/']
Using all grids:  ['http://atlfarm009.mi.infn.it:11121/']
[1] /grid/atlas/users/$USERNAME/lloyd_fast_aod.pool.root
[2] /grid/atlas/users/$USERNAME/lloyd_pythia.pool.root
Choose multiple sources [1-2 (1,2,..) | A{ll}]:
```

Or use the `lcg-lr` (locate replica) command on each e.g:

```
> lcg-lr --vo atlas lfn:lloyd_pythia.pool.root
srm://dcache.gridpp.rl.ac.uk/pnfs/gridpp.rl.ac.uk/data/atlas/generated/
2005-08-01/filee3e4bae1-b3cb-499f-ba3f-7b4f09124feb
```

Complete: 🟨🟨🟨⬜⬜
Responsible: Steve Lloyd
Author: Steve Lloyd 01 Aug 2005
Contributors: Gernot Krobath, Steve Lloyd
Last significantly modified by: Steve Lloyd 12 Oct 2005
Not yet reviewed

4.5 Reading AOD on the Grid

This section describes how to read AOD on the (LCGG) Grid. You will need a Grid Certificate and access to a UI as described in WorkBookStartingGrid and know how to submit jobs to the Grid as described in WorkBookAthenaGrid.

You can either compile and build your code locally and just send object libraries with you job or you can send your source files and build them on the remote site.

Please note that for any serious analysis you should use ATLAS distributed analysis tools as described in AtlasProtected.PhysicsAnalysisWorkBookBatch2. These recipes are just to give you an idea of how things work.

4.5.1 Compiling Locally

Rather than sending your source files and doing the compilation as part of the remote job, it may be simpler and more efficient to compile your code first and then just send the object library with the job.

Build your code locally as described in, for instance, WorkBookArchiveStartingAODAnalysis. Make a copy (or link) of the relevant object libraries i.e.

```
> cp testarea/14.2.23/InstallArea/i686-slc4-gcc34-opt/lib/libUserAnalysis.so .
```

You then need to create two files, the first is the script which you will run, say `athena_anal.sh`, containing:

```
#!/bin/bash
# Script to run Athena Analysis on the Grid

# Get the data

lcg-cp -v --vo atlas lfn:rome.004201.recov10.ZeeJimmy._00018.AOD.pool.root file
    ://`pwd`/Zee_18.pool.root
lcg-cp -v --vo atlas lfn:rome.004201.recov10.ZeeJimmy._00019.AOD.pool.root file
    ://`pwd`/Zee_19.pool.root
ls -lt

source $VO_ATLAS_SW_DIR/software/14.2.23/setup.sh
source $SITEROOT/dist/14.2.23/Control/AthenaRunTime/*/cmt/setup.sh

export LD_LIBRARY_PATH=`pwd`:${LD_LIBRARY_PATH}

athena.py AnalysisSkeleton_jobOptions.py
```

and a second which contains the Job Description Language commands to run the job, say `athena_anal.jdl`:

```
############# Athena AOD Analysis #################
Executable = "athena_anal.sh";
StdOutput = "athena_anal.out";
StdError = "athena_anal.err";
InputSandbox = {"athena_anal.sh","AnalysisSkeleton_jobOptions.py", "
    libUserAnalysis.so"};
OutputSandbox = {"athena_anal.out","athena_anal.err", "AnalysisSkeleton.root", "
    CLIDDBout.txt"};
#InputData = {"lfn:rome.004201.recov10.ZeeJimmy._00018.AOD.pool.root",
#              "lfn:rome.004201.recov10.ZeeJimmy._00019.AOD.pool.root"};
#DataAccessProtocol = {"gsiftp"};
Requirements = Member("VO-atlas-production-14.2.23", other.
    GlueHostApplicationSoftwareRunTimeEnvironment);
##################################################
```

The two lines `InputData` and `DataAccessProtocol` would tell the Grid Resource Broker to submit the job to a computing element close to where this data is physically stored to avoid large data transfers over the network. However if there is no such computing element the job will not be submitted anywhere. In this case a replica should really be made somewhere first. With these lines commented out as above, the job could run anywhere and the data would be copied there.

You can then submit the job to the Grid as normal.

4.5.2 Compiling your own Package Locally

If you have built your own package (say MyNewPackage rather than using AnalysisSkeleton) you have to add a few extra steps. In your input sandbox you need:

```
InputSandbox = {"MyNewPackage.sh","libMyNewPackage.so","MyNewOptions.py","
    MyNewPackage.rootmap",
                                  "MyNewPackageConf.py","MyNewPackage_confDb.py", "
                                      __init__.py"};
```

The .rootmap file can be found in testarea/14.2.23/InstallArea/i686-slc4-gcc34-opt/lib/ (it might be called testarea.rootmap) and the last three can be found in testarea/14.2.23/InstallArea/python/MyNewP
Then in your .sh script that you run on the Grid you need to add:

```
mkdir MyNewPackage
mv MyNewPackageConf.py MyNewPackage
mv MyNewPackage_confDb.py MyNewPackage
mv __init__.py MyNewPackage

export ROOTMAPSEARCHPATH=`pwd`:${ROOTMAPSEARCHPATH}
```

These will get picked up from lines such as these in your JobOptions:

```
from MyNewPackage.MyNewPackageConf import MyAlg
job += MyAlg("MyNewPackage")
```

4.5.3 Compiling Remotely

You need to check out the UserAnalysis package as described in WorkBookRunAthenaHelloWorld#HelloWorldRelease and copy the following files from PhysicsAnalysis/AnalysisCommon/UserAnalysis/UserAnalysis-00-13-06 to your Grid User Interface machine:

```
/cmt/requirements
/cmt/Makefile
/src/components/UserAnalysis_entries.cxx
/src/components/UserAnalysis_load.cxx
/UserAnalysis/AnalysisSkeleton.h
/src/AnalysisSkeleton.cxx
/share/AnalysisSkeleton_jobOptions.py
```

You can edit these files and add other appropriate to your analysis as necessary. You then need to create two files, the first is the script which you will run, say athena_anal.sh, containing:

```
#!/bin/bash
# Script to run Athena Analysis on the Grid

# Get the data

lcg-cp -v --vo atlas lfn:rome.004201.recov10.ZeeJimmy._00018.AOD.pool.root file
    ://`pwd`/Zee_18.pool.root
lcg-cp -v --vo atlas lfn:rome.004201.recov10.ZeeJimmy._00019.AOD.pool.root file
    ://`pwd`/Zee_19.pool.root
ls -lt

source $VO_ATLAS_SW_DIR/software/14.2.23/setup.sh
source $SITEROOT/dist/14.2.23/Control/AthenaRunTime/*/cmt/setup.sh

# Set up the correct directory structure

cmt create UserAnalysis UserAnalysis-00-13-06

mkdir UserAnalysis/UserAnalysis-00-13-06/UserAnalysis
mkdir UserAnalysis/UserAnalysis-00-13-06/src/components

mv AnalysisSkeleton.cxx UserAnalysis/UserAnalysis-00-13-06/src
mv AnalysisSkeleton.h UserAnalysis/UserAnalysis-00-13-06/UserAnalysis
mv UserAnalysis_load.cxx UserAnalysis/UserAnalysis-00-13-06/src/components
mv UserAnalysis_entries.cxx UserAnalysis/UserAnalysis-00-13-06/src/components
mv requirements Makefile UserAnalysis/UserAnalysis-00-13-06/cmt

# Build my code

export CMTPATH=`pwd`:${CMTPATH}

cd UserAnalysis/UserAnalysis-00-13-06/cmt

cmt config
source setup.sh
gmake

# Run

cd ../../..

athena.py AnalysisSkeleton_jobOptions.py
```

and a second which contains the Job Description Language commands to run the job, say `athena_anal.jdl`:

```
############# Athena AOD Analysis #################
Executable = "athena_anal.sh";
StdOutput = "athena_anal.out";
StdError = "athena_anal.err";
InputSandbox = {"athena_anal.sh",
                       "AnalysisSkeleton_jobOptions.py",
                       "../src/AnalysisSkeleton.cxx",
                       "../UserAnalysis/AnalysisSkeleton.h",
                       "../src/components/UserAnalysis_entries.cxx",
                       "../src/components/UserAnalysis_load.cxx",
                       "../cmt/requirements",
                       "../cmt/Makefile"};
OutputSandbox = {"athena_anal.out","athena_anal.err", "AnalysisSkeleton.root", "
    CLIDDBout.txt"};
#InputData = {"lfn:rome.004201.recov10.ZeeJimmy._00018.AOD.pool.root",
#              "lfn:rome.004201.recov10.ZeeJimmy._00019.AOD.pool.root"};
#DataAccessProtocol = {"gsiftp"};
Requirements = Member("VO-atlas-production-14.2.23", other.
    GlueHostApplicationSoftwareRunTimeEnvironment);
#####################################################
```

You can then submit the job to the Grid as normal.

Complete: 🟨🟨⬜
Responsible: Steve Lloyd
Author: Steve Lloyd 19 May 2005
Contributors: Steve Lloyd
Last significantly modified by: Steve Lloyd 21 Feb 2007
Not yet reviewed

4.6 Using Ganga

Complete:
Responsible: Vivek Jain
Author: Steve Lloyd 22 Feb 2007
Contributors: Steve Lloyd
Last significantly modified by: Steve Lloyd 08 May 2007
Not yet reviewed

Chapter 5

Athena Reference

5.1 Athena Options

5.1.1 Command Line Options

Use "`athena.py --help`" to obtain a list of accepted command line options. Of note are:

- `-i`: Run in interactive mode. The athena prompt will be reached after execution of all scripts that were given on the command line.
- `-b`: Run in batch mode. Execute one run of the specified ("theApp.EvtMax") number of events and exit.
- `-l <log level>`: Sets the global log level for the Athena message service as well as for POOL. Acceptable levels are the standard Athena ones (DEBUG, WARNING, ERROR, etc.).
- `-s`: Show the lines of code from the included files as they are being executed. For simple jobOptions this is simple. If the line starts with an A then that line was executed. It also keeps track of the line number of the jobOption that is being executed. eg.

```
-+- 10  92 A if hasattr(runArgs,"outputPixelCalibNtup"):
 -+- 10  93       from InDetRecExample.InDetJobProperties import InDetFlags
 -+- 10  94       InDetFlags.doPixelTrkNtuple.set_Value_and_Lock( True )
 -+- 10  95       from InDetRecExample.InDetKeys import InDetKeys
 -+- 10  96       InDetKeys.trkValidationNtupleName.set_Value_and_Lock( runArgs.
    outputPixelCalibNtup )
```

the if statement fails so the lines following it are not executed as they don't start with an A. For lines starting with a T or C - these are python classes and are harder to interpret.

- `-c "<python statement>"`: Execute the given line of python just before any of the scripts that were specified on the command line. This is useful to set switches that are picked up by the following scripts, to select specific configuration without having to edit them.
  - For instance, `athena.py -c "EvtMax=5" myTopOptions.py` allows you to choose the number of events to analyze on the fly.
  - **However, this works only if EvtMax is commented out in myTopOptions.py**
- `-d [debugger]`: Hook the given debugger to the athena process, just before starting the run (ie., after including all the scripts that were given on the command line). By default, the debugger is gdb, but xdb can be specified for Solaris. See the related topic StartingDebuggerWithAthenaPy.

Further details are available at UsingAthena, as well as at the basic athena and interactive athena tutorials. A complete listing of available flags can be found at RecExCommonFlags.

5.1.2 gmake Options

- `gmake QUIET=1` Quiet mode minimizes output
- `gmake -j4` Compile in parallel in four threads, in general -j . Output messy w/o QUIET=1
- `gmake QUICK=1` Avoid generating the makefiles with dependencies, should only use on a second pass e.g. when one recompiles changes in an existing .h, .cxx file. The dependency makefiles are needed, for example, to force recompilation of any .cxx file that includes a changed .h file.
- `gmake clean` Deletes dependency files, object files to ensure you compile everything again.

Originally taken from https://twiki.cern.ch/twiki/bin/view/Atlas/UsingAthena. gmake options provided by R.D.Schaffer.

Complete: 🟨⬜
Responsible: Steve Lloyd
Author: Edward Moyse 29 Apr 2004
Contributors: Steve Lloyd, Edward Moyse
Last significantly modified by: Steve Lloyd 13 Feb 2007
Not yet reviewed

5.2 Histograms and Ntuples

- This section describes how to use the histogramming and ntuple services in Athena and provides links to example code. The recommended method is to use AtlasProtected.AthenaAwareNTuple (AAN) or Structured Athena-Aware Ntuple (StructuredAthenaAwareNtuple) but simple ROOT histograms can also be used.
  - Main.VivekJain comments "...not sure if these Structured ntuples are still supported..."

5.2.1 Simple ROOT Histograms

In order to create simple ROOT Histograms you need to do the following. In your header (.h) file add:

```
#include "TH1.h"
```

and private members for each histogram variable:

```
private:
   ...
   TH1F* m_h_elecpt;
   TH1F* m_h_eleceta;
   ...
```

In your job options (.py) file include:

```
THistSvc = Service ( "THistSvc" )
THistSvc.Output = ["file1 DATAFILE='myhists.root' OPT='RECREATE'"]
```

The histograms are booked in the `initialize()` method of the algorithm in the C++ (.cxx) file:

```
m_h_elecpt      = new TH1F("elec_pt","pt el",50,0,250.*GeV);
  sc = m_thistSvc->regHist("/file1/Electron/elec_pt",m_h_elecpt);
  m_h_eleceta     = new TH1F("elec_eta","eta el",70,-3.5,3.5);
  sc = m_thistSvc->regHist("/file1/Electron/elec_eta",m_h_eleceta);
```

and filled in the `execute()` method:

```
m_h_elecpt->Fill( (*elecItr)->pt(), 1.);
  m_h_eleceta->Fill( (*elecItr)->eta(), 1.);
```

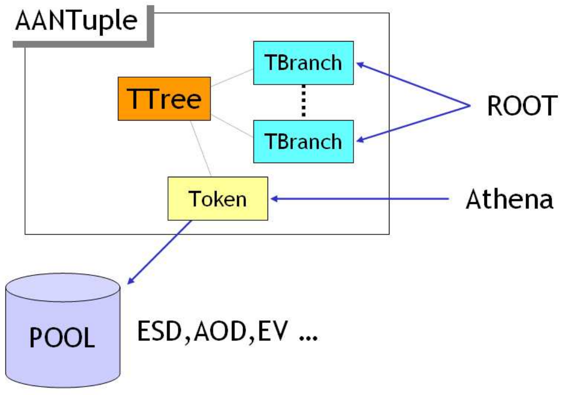
This will store the histograms in the `Electron` sub-directory of the file `myhists.root` (logical file `file1`).

For further details see THistSvc. A working example exists in the AnalysisExamples in `THistSkeleton`.

5.2.2 Athena-Aware NTuple

The AtlasProtected.AthenaAwareNTuple (AANT) is based on the `THistSvc` described above. It allows the user to use his or her output NTuple as an input to another Athena job - for example one could analysis the AAN, pre-select some events, then navigate back to the AOD, ESD or even the Raw data to access the full events: note that the ordinary NTuple cannot be used as an input of Athena jobs (only the Athena-Aware NTuple has this functionality). Thus, the AAN works as the user's version of the event tags for selection and analysis at the same time. A useful introductory talk about the AAN can be found here. A very good working example of the Athena-Aware NTuple is in the UserAnalysis package. The Athena-Aware NTuple is the recommended NTuple for users.

CAVEAT: An algorithm that runs within Athena access information via calls to StoreGate. Information in an Athena-Aware Ntuple is **not** arranged in this way, hence, you cannot simply substitute such an ntuple for an AOD/ESD file in your jobOptions file and expect to run. You need to make other modifications. Examples will be in the Physics Analysis Workbook



Schematic representation of an AANTuple (from Tadashi Maeno).

The AANT has two aspects - a ROOT file with a simple Tree for fast analysis in ROOT and as input to an Athena job with references to events in POOL via a Token.

To add AANT to your job you need to do the following: In your header (.h) file add private members for each variable:

```
private:
   ...
   std::vector<double> * m_aan_eta;
   std::vector<double> * m_aan_pt;
   ...
```

In the initialize() method of the algorithm in the C++ (.cxx) file add:

```
addBranch("ElectronEta",   m_aan_eta);
   addBranch("ElectronPt",    m_aan_pt);

   m_aan_eta->clear();
   m_aan_pt->clear();
```

and in the execute() method:

```
m_aan_eta->push_back((*elecItr)->eta());
    m_aan_pt->push_back((*elecItr)->pt());
```

5.2.3 Reading Histogram or Ntuples Files in Root

This needs to be made more generic and follow the other Workbook examples more

An example of how you can access your histograms and ntuples in root follows: (These examples assume you have created a root ntuple using the HiggsAnal as described in Running an example job at NIKHEF) and you have called your histogram file atlfast1.root and your ntuple file atlfast2.root.

The easiest way to look at your files is to use the ROOT browser.

```
> root

root[0] TBrowser b;
```

To access the root file and its contents from the Root command line you could do the following. For the histogram file:

```
root[0] TFile *f1 = new TFile("atlfast1.root");
root[1] h100 = f1->Get("simple1D/100");
root[2] h100->Draw();
```

For the tree (ntuple):

```
root[3] TFile *f2 = new TFile("atlfast2.root");
root[4] TTree *tree = (TTree *)f2->Get("/Atlfast/2002");
root[5] tree->Draw("H_M");
```

or write a root macro, for example. In a file named readfile.C

```
// Example of how to read a root file and access the histograms or tree
// produced in an Athena job.

// Make tree a global variable so that we can access it outside this function.
// You can then also access it from the command line.
TTree *tree;

readHist(TString filename = "atlfast1.root")
{
  TFile *f = new TFile(filename);
  TH1F *h100 = (TH1F *)f->Get("simple1D/100");
  h100->Draw();
}

readTree(TString filename = "atlfast2.root")
{
  TFile *f = new TFile(filename);
  tree = (TTree *)f->Get("/Atlfast/2002");
  tree->Draw("H_M");

}

// An example of how to access the tree/ntuple variables
eventLoop()
{
  float h_m;
  tree->SetBranchAddress("H_M",&h_m);
  for (int i=0; i < tree->GetEntries(); i++){
    tree->GetEntry(i);
    cout << h_m << endl;
  }
}
```

Then from root (assuming the above file is called `readfile.C`) you can do the something like the following:

```
> root

root[0] .L readfile.C
root[1] readHist("atlfast1.root");
root[2] readTree("atlfast2.root");
root[3] tree->Draw("H_M","H_PT>10");
root[4] eventLoop();
```

5.2.4 Structured Athena-Aware NTuple

Structured Athena-Aware NTuples or simply AtlasProtected.StructuredAthenaAwareNtuple are AtlasProtected.AthenaAwareNTuple where some or all of the branches contain structured objects such as the AOD Electron. The AtlasProtected.StructuredAthenaAwareNtuple is a ROOT Tuple where instead of saving simple variables such as "pt", "eta", "phi", you save a complex, structured object, for instance the Calorimeter Clusters: the complex objects contain - if relevant - its own 4-momentum and additional information in a compact, structured way. The AtlasProtected.StructuredAthenaAwareNtuple allows one to have ROOT accessibility for the ESD and the AOD; the data structure is compact and easy to browse; the AtlasProtected.StructuredAthenaAwareNtuple provides the ability to port back to Athena pieces of codes developed within ROOT. The AtlasProtected.StructuredAthenaAwareNtuple is also Athena-Aware in the sense that one can make event selections on the AtlasProtected.StructuredAthenaAwareNtuple just as we do on with the TAG, and access the AOD, ESD or even Raw Data of the selected events. To get started with examples, follow this link: AtlasProtected.StructuredAthenaAwareNtuple.

WorkBookArchiveHistograms - Histograms and Ntuples (Archived Version)

Originally taken from `http://www.nikhef.nl/pub/experiments/atlas/software/www/AthenaHistogramming.`

html

Complete: 🟨
Responsible: Steve Lloyd
Author: Grant Gorfine 26 Jul 2005
Contributors: Ketevi Assamagan, Grant Gorfine, Steve Lloyd
Last significantly modified by: Ketevi Assamagan 19 Dec 2005
Not yet reviewed

5.3 Accessing Trigger Information

5.3.1 The ATLAS Trigger System

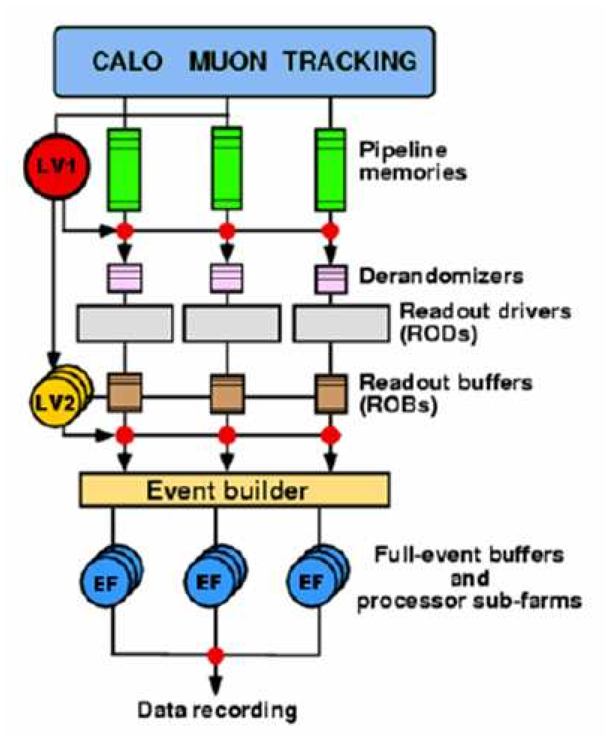
The ATLAS Trigger System is designed to reduce the 40 MHz bunch crossing rate to ∼200 Hz for offline storage and processing e.g. ∼5 events per million crossings! It is essential that the correct events are written out. There are three trigger levels:

$ **Level 1** Hardware based (FPGA/ASIC) with coarse granularity detector data. Uses only the calorimeter and muons with a latency 2.2 micros (on-detector buffer) and an output rate ∼75 kHz.

$ **Level 2** Software based but only detector sub-regions (Regions of Interest) seeded by Level 1 are processed. The full detector granularity in RoIs is used with fast tracking and calorimetry. The average execution time ∼10 ms and an Output rate ∼1 kHz.

$ **Event Filter (EF)** Seeded by level 2 uses the full detector granularity with potential full event access to offline algorithms. The average execution time ∼1 s with an output rate ∼200 Hz.

Level 2 and the Event Filter are together known as the **High-Level Trigger** (HLT).



Schematic representation of the ATLAS Trigger System.

The **Trigger Menu** contains a list of all active **Trigger Signatures**. Each signature is designed to look for a particlular occurance in the detector.Examples are:

- At Level 1 - EM25i means Electromagnetic, Pt > 25 GeV, Isolated.
- At the HLT - e25i means Electron, Pt > 25 GeV, Isolated.

Other possibilities are MU, HA (Hadronic), Tau, J (Jet), E (Energy), xE (Missing Energy) etc.

5.3.2 Accessing the Trigger Information

Useful information for making a trigger aware analysis can be found in the Trigger User Pages, which are currently under construction. For 12.0.4 there are various tutorials available which you might want to check out. The tutorial are collected in the Trigger Tutorial Page

The 'User Interface' of the Trigger code is the TriggerDecision class. This is intended to provide users with a (simple) interface to finding the decision of the trigger for each event. It should provide a list of which trigger signatures were running at each trigger level, and which were satisfied by the event. It should also provide methods which return the overall result (accept/fail) from each trigger level and for the whole trigger. For each event, a TriggerDecision object is produced by an Algorithm derived class TriggerDecisionMaker.

The TriggerDecision will contain the trigger menu items as specified by the configuration at reconstruction time. In 12.0.4 the menu used in the official production is called CSC-03. It's definition can be found here. Note, in the current Trigger Menu you find various physics trigger for different luminosities (10^{33} and $10^{31)}$, triggers which will be pre-scaled but no pre-scale factors have been applied. Mixed triggers are not available in the current trigger set-up and will be only correctly implemented in release 13. In the moment you have to create mixed triggers by hand for example by asking is mu10 fulfilled AND is e15i fullfilled.

To use it just retrieve it from StoreGate in your code, ask if a given signature was defined and ask if each event passed your favourite signature. Note that the trigger decision stores the event-level decision, but the processing is RoI based.

Note: The Trigger decision has been available since 11.0.5 but prior to release 12.0.3 it was included in the ESD but not the AOD.

TriggerDecision gives full details but basically to access the trigger from AOD. If you use the release 12.0.4 you need to do the following:

In cmt/requirements

Add:

```
use  TrigSteeringEvent    TrigSteeringEvent-00-*   Trigger/TrigEvent
```

In AnalysisSkeleton.cxx

Near the beginning add:

```
#include "TrigSteeringEvent/TriggerDecision.h"
```

In `AnalysisSkeleton::CBNT_execute()` you can use the code below:

Receive all Trigger objects from StoreGate:

```
MsgStream mLog(msgSvc(), name());
  mLog << MSG::INFO << "Trying to retrieve TriggerDecision objects" << endreq;

  const TriggerDecision* trigDec = 0;
  StatusCode sc=m_storeGate->retrieve( trigDec, "MyTriggerDecision");
  if( sc.isFailure()  ||  !trigDec ) {
    mLog << MSG::WARNING
         << "No TriggerDecision found in TDS"
         << endreq;
    return StatusCode::SUCCESS;
  }

  mLog << MSG::INFO << "TriggerDecision retrieved" << endreq;
```

The above will work if you want to read the original TriggerDecision from a ESD/AOD, which has the key MyTriggerDecision. In case you have re-run the hypothesis in your job, a new TriggerDecision will be created. This time the key will be !MyTriggerDecision+, the next one !MyTriggerDecision++ etc. To ensure you take the latest instance of TriggerDecision you can use

```
// look for TriggerDecision and pick the latest (in case of trigegr re-run)
  std::string new_TD = m_TriggerDecisionKey;
  std::string lastTD = m_TriggerDecisionKey;
  bool old_td  = false;
  do {
    if (m_storeGate->contains<TriggerDecision>( new_TD )) {
      log << MSG::DEBUG << "TriggerDecision with key " << new_TD
     << " found in StoreGate; looking for " << new_TD+"+" << endreq;
      old_td = true;           // flag for do...while loop
      lastTD = new_TD;         // keep last TD key before updating new
      new_TD = new_TD + "+";   // update StoreGate key
    } else {
      old_td = false;           // to stop loop
      log << MSG::DEBUG << "No TriggerDecision with key " << new_TD
     << " found in StoreGate; using key " << lastTD  << endreq;
    }
  } while (old_td);

  // retrieve latest TriggerDecision
  const TriggerDecision* trigDec = NULL;
  sc = m_storeGate->retrieve(trigDec, lastTD);
  if ( sc.isFailure() ) {
    if(outputLevel <= MSG::WARNING) {
      log << MSG::WARNING << "Failed to retrieve TriggerDecision" << endreq;
    }
    return StatusCode::SUCCESS;
  }
```

To access the Trigger decision:

```
mLog << MSG::INFO << "Looking at TriggerDecision " << i << endreq;
      if (trigDec->isTriggerPassed()) {
           mLog << MSG::INFO << "Trigger passed" << endreq;
      } else {
          mLog << MSG::INFO << "Trigger failed" << endreq;
      }
      // check if given signature e.g. e25i passed, note L1==1, L2==2, EF==3
      if (trigDec->isDefined("L2_e25i",2) && trigDec->isTriggered("L2_e25i")) {
         mLog << MSG::INFO << " e25i passed" << endreq;
      }

      mLog << MSG::INFO << "Printing Trigger Decision " << i << endreq;
      trigDec->print();
```

You should get something like this:

```
TriggerDecision: no of TriggerItems defined in this run 80
TriggerDecision: Level 1:
TriggerDecision...   INFO TriggerDecision: L1 Trigger Item L1_2EM15  did not
   pass
TriggerDecision...   INFO TriggerDecision: L1 Trigger Item L1_2EM15I  did not
   pass
TriggerDecision...   INFO TriggerDecision: L1 Trigger Item L1_2J45  did not pass
...
TriggerDecision: Level 2:
TriggerDecision...   INFO TriggerDecision: L2 Trigger Item L2_Z(e10e10)  did not
    pass
TriggerDecision...   INFO TriggerDecision: L2 Trigger Item L2_b35  did not pass
TriggerDecision...   INFO TriggerDecision: L2 Trigger Item L2_e10  did not pass
TriggerDecision...   INFO TriggerDecision: L2 Trigger Item L2_e10L2_e10  did not
    pass
...
TriggerDecision: Event Filter:
TriggerDecision...   INFO TriggerDecision: EF Trigger Item EF_MuonTRTExt_mu6l
   did not pass
TriggerDecision...   INFO TriggerDecision: EF Trigger Item EF_b35  did not pass
TriggerDecision...   INFO TriggerDecision: EF Trigger Item EF_e10  did not pass
...
```

Note: The selection cuts used to select the events are not optimal in 12.0.4 as the newer simulated datasets were produced with updated material distributions etc. The efficiencies and rate still need re-optimisation for these datasets and the selections are rather indicative at this point. Updated optimisations will be provided in time and can be used by re-running the trigger hypothesis as for example shown in the tutorial here.

Originally taken from a talk by Ricardo Goncalo at the UK ATLAS Physics Meeting in Durham 2006.

Complete: ▰▰▰▰▱
Responsible: Steve Lloyd
Author: Ricardo Goncalo 07 Sep 2006
Contributors: Ricardo Goncalo, Steve Lloyd, Monika Wielers
Last significantly modified by: Monika Wielers 19 Jan 2007
Not yet reviewed

Chapter 6

Software Infrastructure

6.1 Installing ATLAS Software

Most users should not have to install ATLAS software themselves as it will probably already be installed centrally at most sites. However, some users may wish to install different versions or install ATLAS software on their laptop, or at home. This section describes how to simply install the latest production release of the software locally as a **kit**. Full details can be found at AtlasSoftwareInstallInstructionsAdvanced. **Note**: These instructions are only guaranteed to work on the ATLAS reference platform - currently **Scientific Linux 4** or equivalent.

If you have problems please ask for help via Hypernews: atlas-releaseKitProblem or email hn-atlas-releaseKitProblem@cern.ch. Bug reports and feature requests are filed in Savannah: atlas-infra

Any other feedback should also be sent to Hypernews: atlas-releaseKitProblem.

Announcements about releases (such as availability and patches) are made in Hypernews: atlas-releaseKitAnnounce

6.1.1 Installing Pacman

Before to install Pacman please note that it is available on AFS so you can use it directly

```
source /afs/cern.ch/atlas/software/pacman/pacman-latest/setup.[c]sh
```

If you really need to install Pacman you can download it from Pacman Headquarters. You should get a tar file called `pacman-latest.tar.gz`. You don't have to worry about having the right version of Python. If necessary, Pacman will build a local Python installation for you and tell you what to do.

Start with a **clean shell** without having sourced any ATLAS or CMT setup scripts previously (either explicitly or in your login scripts).

`cd` to the directory you want to use (needs **lots** of disk space) and then do:

```
> tar -zxf pacman-latest.tar.gz
> cd pacman-*
> source setup.sh [or .csh]
```

Check that Pacman is OK and what versions are available by doing:

```
> cd .. [or somewhere else]
> pacman -lc am-CERN
```

6.1.2 Installing the ATLAS Software

Create a directory to hold the release:

```
> mkdir 14.2.23
> cd 14.2.23
```

and then do

```
> pacman -get am-CERN:14.2.23
```

Replacing `am-CERN` with the closest mirror from:

- `am-CERN` (CERN/Tier 0)
- `am-BNL` (Brookhaven National Laboratory/US Tier 1)
- `am-RAL` (Rutherford Laboratory/UK Tier 1)
- `am-BU` (Boston University/Northeast US Tier 2)
- `am-IHEP` (IHEP/China)
- `am-IU` (Indiana University/Midwest US Tier 2)
- `am-UM` (University of Michigan/Great Lakes US Tier 2)

and answer questions like:

```
Do you want to add [http://cern.ch/atlas-computing/links/kitsDirectory/projects/
    cache]
to [trusted.caches]? (y or n): y
```

While unpacking the distribution archives, tar might find overlapping files from different packages. In this case, tar asks for an interactive confirmation of the overwrite, . If you think you can safely overwrite the files, use the pacman option

```
-allow tar-overwrite
```

e.g.:

```
> pacman -allow tar-overwrite -get am-CERN:14.2.23
```

Then eventually you should see something like:

```
> ls
CMT/                trusted.caches     pacman/       AtlasSettings/
 AtlasLogin/        sw/                DBRelease/    atlas/
 project/           usr/               LCGCMT/       Gaudi/
 AtlasCore/         AtlasConditions/   AtlasEvent/   AtlasReconstruction/
 AtlasTrigger/      AtlasAnalysis/     geant4/       external/
 AtlasSimulation/   AtlasProduction/   cmtsite/      AtlasOffline/
KV-14.2.23/          setup.sh           setup.csh     o..pacman..o/
```

This procedure will test all pre-conditions so you don't have to worry about gcc, Python, Pacman versions, OS, processor, Unix shell, post-installation patches, etc. It also installs the database replica automatically. If something is wrong, you will get an error message.

To install the same release with kit validation, do the following instead.

```
> pacman -get am-CERN:14.2.23+KV
```

If you see any messages from the kit validation other than OK and PASSED there is a problem and you should seek help.

Note that the time taken for complete installation varies depending on factors such as your network bandwidth and how close you are to the chosen mirror. I can take up to a couple of hours. The AFS mirror can be very slow if you have a small cache.

Use `% pacman -lc -d patch version` to check the status and contents of your installation and `% pacman -lc am-CERN` to check the status of any of the mirrors and to see what releases are currently available. If you have problems or comments, post something to the hypernews.

6.1.3 Kit Validation

If you include `+KV` when you installed the kit it will be automatically validated.

You can also run KitValidation after installing a release by this command:

```
> pacman -get am-CERN:KV-14.2.23
```

You should get something like this:

```
##################################################################
##          Atlas Distribution Kit Validation Suite             ##
##                    29-04-2008  v1.9.16-1                      ##
##                                                               ##
## Alessandro De Salvo <Alessandro.DeSalvo@roma1.infn.it> ##
##################################################################
 Testing AtlasProduction 14.2.0
 athena executable                              [PASSED]
 athena shared libs                             [PASSED]
 Release shared libraries                       [PASSED]
 Release Simple Checks                          [  OK  ]
 Athena Hello World                             [  OK  ]
 MooEvent compilation                           [  OK  ]
 /opt/atlas/14.2.0/KV-14.2.0/tmp
DB Release consistency check                    [  OK  ]

#######################################################
##   AtlasProduction 14.2.0 Validation [  OK  ]
#######################################################
```

Individual tests are PASSED/FAILED and if a group of tests all pass then it shows OK otherwise FAILED. You can compare your output with the results of the centrally run tests by going to the Release Status Page, clicking on the relevant release in the Status column and then clicking on "Kit Validation results".

6.1.4 Using the Installed Kit

You should now be able to use the installed kit as described WorkBookSetAccount. In the examples shown above the directory structure looks like:

```
/disk/data73/lloyd/pacman-3.25
/disk/data73/lloyd/14.2.23/...
/disk/data73/lloyd/14.2.23/AtlasCore
/disk/data73/lloyd/14.2.23/AtlasCore/2.0.1/...
/disk/data73/lloyd/14.2.23/AtlasCore/2.0.1/Control/...
```

Hence `path_to_kit` is `/disk/data73/lloyd/14.2.23/`

Further Details

For more details, see

- Using Pacman
- Distribution Kit Paths To Avoid
- Updating an existing installation
- Making a multi-release installation
- Making your own mirror
- Handling tar-overwrites
- Installing projects
- Using project kits
- Installing individual projects, src or debug builds
- Installing nightly software builds
- Advice on unsupported platforms
- Testing if a computer is ATLAS-ready
- Installing the database replica by hand
- SLC4 compatibility RPM's for 32 bit

Previous Releases

- 13.0.30
- 12.0.6
- 11.0.5

Originally taken from InstallingAtlasSoftware and DraftNewInstallForWB

Complete: 🟩
Responsible: Steve Lloyd
Author: Christian Arnault 06 Sep 2004
Contributors: Christian Arnault, Marcello B, Simon George, Steve Lloyd, Saul Youssef
Last significantly modified by: Marcello B 22 Jan 2008
Last reviewed by: Saul Youssef 20 Jul 2006

6.2 Don Quijote 2 (DQ2)

DQ2 (Don Quijote 2) is the ATLAS Experiment Data Management System. Its goal is to integrate all Grid data management services used by the ATLAS Experiment, providing production managers and physicists access to file-resident event data, implementing the data flow as defined by the ATLAS Computing Model. For full details see the Distributed Data Management Documentation. You need a Grid Certificate to use the tool.

6.2.1 How do I use DQ2?

Please go directly to the DQ2ClientsHowTo.

It has all the updated information and is the only definitive source on how to use DQ2. This is kept updated directly from the developers.

Please do not start new wiki-pages on how to use DQ2 but instead link to that page. Thank you!

Complete: 🟩
Responsible: Steve Lloyd
Author: Mario Lassnig 06 Apr 2009
Contributors: Mario Lassnig, Steve Lloyd
Last significantly modified by: Steve Lloyd 08 May 2007
Not yet reviewed

6.3 The Atlantis Event Display

Atlantis is a Java-based application for visualising ATLAS events. Atlantis development is guided by three principles:

1 Atlantis is fast (operations should take no longer than a second).
2 Atlantis is used intuitively (easy to use, easy to interpret).
3 Atlantis is used for complete ATLAS events (all subsystems displayed).

Atlantis takes as input the XML output files from JiveXML. Geometry files contain detector layout and event files contain physics information.

6.3.1 Creating XML files using JiveXML

JiveXML is an Athena package that contains algorithms to convert event Data to XML files. The XML files can then be read in by the Atlantis Event Display. Both fully reconstructed and fast simulated events (Atlfast) can be converted to XML and viewed with Atlantis.

The procedure is similar to that for reconstructing events since RecExCommon has a special flag, `doJiveXML` (=False by default), for running JiveXML.

Go to your run directory such as:

```
> cd 14.2.23/PhysicsAnalysis/AnalysisCommon/UserAnalysis/run
```

and create a Job Options file such as `myJiveXMLOptions.py` with the following lines in it, changing `PoolRDOInput` appropriately:

```
##############################################################
# Detector description
DetDescrVersion = 'ATLAS-CSC-02-02-00'

# suppress the production of ntuple and histogram files
doCBNT = False
doHist = False

EvtMax=-1
```

```

PoolRDOInput=["g4digi.pool.root"]

doJiveXML=True
##################################################################
```

Now you can run athena:

```
> athena.py myJiveXMLOptions.py RecExCommon_topOptions.py > athena_xml.out
```

You should have a a number of XML files (this was Z0→e+e- events) such as these:

```
-rw-r--r--   1 lloyd   hep      5696195 May 13 13:08 JiveXML_1337_00001.xml
-rw-r--r--   1 lloyd   hep      5590310 May 13 13:07 JiveXML_1337_00000.xml
```

Running on ESD or AOD

If you want to use an ESD as the input file, please follow the instructions here.

Simplified running on AOD

The above procedures produces XML files from RDO, ESD or AOD with the maximum possible content at the cost of a bit more complicated setup. If you want to run on AOD in a simpler mode with access only to the leptons, jets etc you can do the following: Assuming you are using `AnalysisSkeleton` as described in WorkBookReadingAOD then after the line

```
AnalysisSkeleton = Algorithm( "AnalysisSkeleton" )
```

add:

```
include( "JiveXML/JiveXML_jobOptionBase.py" )
include( "JiveXML/DataTypes_AOD.py" )
include( "JiveXML/DataTypes_Reco.py" )
include( "JiveXML/DataTypes_Trig.py" )
```

and then run athena as before:

```
> athena.py AnalysisSkeleton_jobOptions.py > athena_anal.out
```

You should get something like this:

```
-rw-r--r--  1 lloyd hep  44924 Oct  4 11:35 JiveXML_0_00002.xml
-rw-r--r--  1 lloyd hep  79005 Oct  4 11:35 JiveXML_0_00001.xml
-rw-r--r--  1 lloyd hep  73647 Oct  4 11:35 JiveXML_0_00000.xml
```

6.3.2 Obtaining and Running Atlantis

Atlantis can be obtained from the download area of the Atlantis Website or can be run directly from AFS `/afs/cern.ch/atlas/project/Atlantis/current`. You need to have Java version 1.4 or higher installed to run Atlantis.

If you are using a remote kit then in principle you can use the version of Atlantis that comes with the kit, replacing `AFS-directory` below with `/path_to_kit/AtlasAnalysis/*/InstallArea/share/AtlantisJava`. Note however that the version that ships with release 12.0.3 won't read 12.0.3 AOD and you will have to download the latest version or use the AFS version.

The general command to run Atlantis is (replace `JAVA-version`, `AFS-directory` and `tag` by appropriate values in the command line):

```
> JAVA-version/bin/java  -jar  AFS-directory/atlantis.jar  tag
```

To run with Fast Simulation events use the `tag Fast`.

In Windows double-click on the Atlantis icon / filename (assuming `jar` files are correctly associated with `javaw`).

Two windows should appear: the Atlantis Canvas and the Atlantis GUI. The Canvas shows a simplified ATLAS detector geometry in the x-y plane with physics signatures from a simulated event. The Atlantis GUI contains various tools for manipulating the event display.

Note: There is a problem with Java on the recent !MacOS-X (Tiger, 10.4.0), which causes a very slow performance of Atlantis. You should use Java Development Kit (JDK) 1.5, with the following startup-line:

```
> JAVA-version/bin/java -Dapple.awt.graphics.UseQuartz=false -jar  AFS-directory
    /atlantis.jar  tag
```

6.3.3 Using Atlantis

Reading Events

Click on the **File** menu and you will see five options. The first three are for reading events from local disk, a URL or a server. Select **Read Event ...** and select the XML file of your first event. You can navigate through the events using **Next** and **Prev**.

The Window Control

The x-y projection is displayed in the Atlantis Canvas by default. There are a number of other different displays immediately available which are accessible via the Window Control. The Window Control contains representations of the Canvas subdivided up in different ways. Click on the subdivision 6, for example, and one ninth of the Canvas is covered with a display of calorimeter cells and extrapolated track positions in the eta-phi projection. Click around to see what views are available

If you would like to see the information from one subdivision displayed with the layout of another, you can use the drag-and-drop facility. For example, click and hold on subdivision 6, then move the cursor over to S. The Inner Detector display is now shown in a much larger region of the Canvas.

To return the Canvas to its original state, just click on Reset in the Menu Bar.

Help

Atlantis has a very extensive help facility. You can access this help by right-clicking on most parts of the Atlantis GUI. A window will appear containing help information relevant to the area of the GUI being pointed to.

Right-click on an unoccupied part of the Window Control. A help window should appear, entitled The Window Control. Right-clicking is sometimes alternatively used to list available functions for the item being pointed to. This can be seen by right-clicking on an occupied part of the Window Control, or any part of the Canvas.

If the Help window is already open and you right-click somewhere else, the information in the Help window changes rather than a new Help window being created. This avoids desktop congestion.

Zoom, Move, Rotate

By default, the Atlantis GUI starts off in Zoom, Move, Rotate (ZMR) mode. The ZMR tab should be highlighted in the Interactions Control. If it isn't, click on the ZMR tab now. Also, click Reset to return the Canvas to its original state.

Pick a point on the Atlantis Canvas (not the red dot in the middle!) and click on it. Holding the click, move your pointer to somewhere else on the Canvas. The display will perform a zoom by forcing the point you originally clicked on the follow your mouse pointer.

Holding down `m` on the keyboard, do the same thing again. This time, the display moves instead of zooming. Similarly, holding down `r` while clicking and pointing will cause the display to rotate.

When holding down a key while using the mouse adds functionality, this is called a 'Mouse Modifier'. To see a list of available Mouse Modifiers for the Interaction you are currently using, select Help → Mouse Modifiers. The list for the ZMR tab includes C - modify Central point. Try this out by holding down c and clicking anywhere on the Canvas. The big red dot in the middle should move to the point you have clicked. This red dot is the focus of the zoom facility. Try zooming again with a changed focus.

Rubberband

Click on the Rubberband tab. You are now in rubberband mode and you may have noticed a drop-down menu appear immediately below the tab. This should currently say Rectangle. Rubberband mode allows you to select a part of the Canvas to inspect more closely. By clicking and dragging on the Canvas, make a rectangular selection. A menu will appear next to the rectangle. By clicking Zoom, the selected portion of the display should now fill the entire Canvas.

Make another rectangular selection and this time, click on Zoom and drag it over to a window or subwindow in the Window Control. The selection should now be displayed in a smaller part of the Canvas

Rectangle is just one of the options in the drop-down menu. You can play around with the other options to see what selection shapes are available

After performing a rubberband zoom, right-clicking in the Canvas allows the user to 'unzoom'

Inspecting the Event

On loading an event, the Output Display shows a summary of the objects found by reconstruction. The Display indicates the number of tracks from the 'iPatRec' algorithm and from the 'xKalman' algorithm. By default, iPatRec tracks are displayed and you should be able to see them.

To change this, click on the Data tab in the Parameters Control. There is a list of algorithms and subdetectors, each with a corresponding tick-box. 'iPat' currently has a tick and 'xKal' does not. To see the xKalman tracks, click on both tick-boxes so that only 'xKal' shows a tick. The Canvas should now have updated with the desired tracks.

Go back to showing iPatRec tracks.

Associated Hits

Finally, we will use Atlantis to show the association between tracks and hits. Click on the ID tab in the Parameters Control. This allows control of display items relating to the Inner Detector. There are some vertical tabs at the side, click on iPat. The controls for tracks from the iPatRec algorithm are now displayed. Where it says Color Function, there is a drop-down menu displaying Constant. Click on the drop-down menu and select Index. You should now see that the two tracks displayed in the Canvas are different colours.

To see which silicon hits are associated to the tracks, click on S3D. This is the control information for silicon hits. There is a drop-down menu labelled Color Function. Click on the menu and select Reconstructed Track. Looking back at the Canvas, you should now see that associated hits are displayed in the same colour as their track. You can do exactly the same thing in the TRT tab to see associated TRT hits.

Previous Releases

- 13.0.30
- 12.0.6
- 11.0.5
- 10.0.1

Originally taken from `http://www.hep.man.ac.uk/u/sdean/atlantis/tutorial/index.html`

Complete: 🟨🟨🟨⬜⬜
Responsible: Steve Lloyd
Author: Steve Lloyd 19 Sep 2006
Contributors: Se Boeser, Simon Dean, Andrew Hamilton, Steve Lloyd, Juergen Thomas
Last significantly modified by: Simon Dean 08 Feb 2005
Last reviewed by: Andrew Hamilton 18 Sep 2006

6.4 Using the CASTOR tape storage

If you store large data files in your private space, you will quickly fill it up. You are therefore provided with a much larger storage space in a tape system, known as CASTOR. This space is backed up, unlike scratch space. Jobs can directly access files stored in CASTOR, and can write to it, but a user cannot open a file from CASTOR: instead, the files must be copied into a local space, and then edited. In addition, you cannot use "tab completion" or "wildcards" with CASTOR - all paths must be typed in explicitly. However, it is possible to move, list, copy and delete files within the CASTOR filesystem.

Also be aware that, as the system involves the use of robotic machinery for reading the tapes, long delays can occur if many users are accessing it.

The castor file structure is:

```
/castor/cern.ch/user/<letter>/<userID>/your_directories
```

The user ID is the same as your normal CERN ID.

More recent information can be found at CastorDefaultPoolRestrictions.

6.4.1 Castor commands

The castor commands are:

- Copy a file from a local directory to CASTOR:

```
rfcp MyFileName /castor/cern.ch/user/<letter>/<UID>/MyDirectory/MyFileName
```

- Copy a file from CASTOR to a local directory:

```
rfcp /castor/cern.ch/user/<letter>/<UID>/MyDirectory/MyFileName MyFileName
```

- Copy a file from one CASTOR location to another:

```
rfcp /castor/cern.ch/user/<letter>/<UID>/MyDirectory1/MyFileName1
     /castor/cern.ch/<letter>/<UID>/MyDirectory2/MyFileName2
```

- Delete a file from CASTOR:

```
rfrm /castor/cern.ch/user/<letter>/<UID>/MyDirectory/MyFileName
```

- Create a directory in CASTOR:

```
rfmkdir /castor/cern.ch/user/<letter>/<UID>/MyNewDirectory/
```

- Move a file in CASTOR:

```
rfrename /castor/cern.ch/user/<letter>/<UID>/MyDirectory/MyOldFileName
         /castor/cern.ch/<letter>/<UID>/MyDirectory/MyNewFileName
```

- List the contents of a CASTOR directory:

```
rfdir /castor/cern.ch/user/<letter>/<UID>/MyDirectory
```

6.4.2 Access CASTOR from a running job:

```
rfio:/castor/cern.ch/user/<letter>/<UID>/MyDirectory/MyFileName
```

still works but it has been replaced by

```
castor:/castor/cern.ch/user/<letter>/<UID>/MyDirectory/MyFileName
```

If you want to read several separate files, for example from a Python file to run within Athena, concatenate them as follows:

```
EventSelector.InputCollections = [
    "castor:/castor/cern.ch/user/<letter>/<UID>/MyDirectory/MyFileName1",
    "castor:/castor/cern.ch/user/<letter>/<UID>/MyDirectory/MyFileName2",
    "etc..."
]
```

Sept 2007: Comment from non-expert: I do not think "castor:/castor/..." works anymore and results the misleading error "unable to load libReaddCache"(!). Use "/castor/..." instead.

This is not an exhaustive list of commands: see here for the full list.

6.4.3 Troubleshooting

- http://cern.ch/castor/faq.htm
- FIOgroup.UsersTransfers

Castor commands

If you see this type of problem:

```
> rfcp  /castor/cern.ch/my_file  .
castor/cern.ch/my_file : No such file or directory
```

This might be caused by getting the LCG rather than the 'castor' version of /usr/bin/rfcp from your PATH. The solutions to this problem are:

a 'fix' the path and continue to use 'rfcp' from /usr/bin or

b Always use /usr/bin/rfcp

within athena

If you get errors by accessing castor within athena;

- read this: https://hypernews.cern.ch/HyperNews/Atlas/get/releaseKitProblem/92/2.html

access too slow / timeout

some good explanations:

- https://hypernews.cern.ch/HyperNews/Atlas/get/cernCompAnnounce/161.html
- https://hypernews.cern.ch/HyperNews/Atlas/get/Prelimbugs/640/1/2/1/1.html

To see the request queue to castor

- https://lemonweb.cern.ch/lrf-castor/

access from your private machine

If you cannot access castor from your private machine at cern, have a look at: CastorOnPrivateMachines

---

Complete: 🟨🟨🟨🟨⬜
Responsible: Steve Lloyd
Author: James Catmore 03 Feb 2005
Contributors: James Catmore, Chris Collins Tooth, Steve Lloyd, Ikuo Ueda
Last significantly modified by: Ikuo Ueda 01 Nov 2006
Last reviewed by: Steve Lloyd 25 Feb 2005

6.5 Viewing ATLAS Code

Different views of the code serve different purposes:</p>

6.5.1 Installed code

The built code is available on AFS for each project and release. The area contains the code (source, libraries, etc.), and the InstallArea against which you normally link when running a job at CERN. The directory structure is "enhanced" compared to the CVS repository (see below) with information about release number and compiler used. The latest release of each project can be accessed in the directory latest.

```
/afs/cern.ch/atlas/software/builds/
/afs/cern.ch/atlas/software/builds/<project>/latest/<package>/
```

More ¡em¿links to the release area¡/em¿ are provided for convenient web access; the corresponding links are maintained also as TWiki variables for use in TWiki documents.

6.5.2 ViewVC

The CVS repository is the central repository for ATLAS software. You can browse with ViewVC the containers, their packages and files. For experts details of the CVS operations (date, version number, etc.) are displayed and tools help to compare different revisions. If you are new to ATLAS software, scan through one of the packages, e.g. Control/AthenaExamples to see the structure of the software packages.

6.5.3 RSS feeds for ATLAS CVS

If you would like to get information about the latest commited changes inside CVS for a particular package please check the information at RSS feeds for ATLAS CVS

6.5.4 LXR

LXR is similar to ViewVC, but has a powerful search tool. If you need to find, where a certain class or parameter is used, this is the tool to use. Both Athena and Gaudi packages are accessible. For details consult the BNL LXR page or see WorkBookLxr.

6.5.5 Doxygen

Doxygen is used in ATLAS to document the code. Doxygen extracts documentation from comments in the source C++ and Python code, and analyses the code for dependencies. A description of the classes and their dependencies is generated and depicted in UML diagrams. Navigation back to the code is provided. Per package, a mainpage provides a description of the package and its interface classes. Doxygen documentation is generated for all releases and nightly builds. Doxygen documentation is very useful if you have to work through details of the code.

Developers should follow the guidelines for writing Doxygen documentation for C++ code and Python scripts.

6.5.6 Doxygen view of Packages (AMI)

AMI provides Doxy doc, and also CVS, LXR access to different packages and releases (later than 11.2.0); graphical display of dependencies of projects and selection of projects. To get a list of all ATLAS packages, click on View all projects packages. By selecting Apply software domain filter on, one can view packages for selected domains.

6.5.7 Doxygen View

Doxygen View (Torre Wenaus) provides Doxy doc, and also CVS, LXR access to different packages and releases (starting from 11.5.0). This is another way of getting access to the Doxygen documentation. The list of external packages and their versions is also shown. If a package has no file html/index.html then no link to the doxy doc is displayed.

6.5.8 Doxygen of Classes - latest nightly release

The Doxygen view described above provides easy access to classes in a package, but it does not allow an overall search for classes. As this was felt to be very useful, a page was provided, which lists all classes of the latest nightly release. This page is not available when the nightly build failed. There is also a page of the classes from the last stable release here

---

Originally taken from static web pages written by Main.TraudlKozanecki .

Complete: ▮▮▮▮▯
Responsible: Steve Lloyd
Author: Traudl Kozanecki 29 Aug 2007
Contributors: Traudl Kozanecki, Steve Lloyd
Last significantly modified by: Steve Lloyd 06 Aug 2008
Not yet reviewed

6.6 !! Indexing ATLAS Software

6.6.1 Introduction

There is a very powerful web-based tool, LXR, that allows you to cross-reference ATLAS software. You can search for a particular pattern of characters or for a specific file.
For instance, if you want to know where a certain variable (in a AOD/ESD or ntuple file) is filled, you can do so with LXR. Of course, if the variable name is very common, then you will get lots of hits.
Another very nice feature is that when you are browsing a file, class names and other declarations are live links, so by clicking on them, you can also explore those files. It is an easy way to navigate ATLAS software.
It is very straightforward to use

TIPS:

- Be sure to read the **Caveats** at the bottom of the LXR main page for tips on searching
- You can browse the latest nightly or the latest stable release. If you do not see the latest stable release, you will have to request that it be "loaded"
  - Send e-mail using the link at the bottom of the LXR mainpage
- Unlike the CVS repository (that is web-viewable), on **LXR** you can only view one version of the software; the one that corresponds to that release
- It is recommended that after you choose the nightly/stable-release option, you choose the "general search" option.
  - Two fillable fields then appear in the window: "Files named" and "Containing".

6.6.2 Example 1

- In the example on accessing MC information in the Physics Analysis Workbook, we mention a method, **getDaughters()**, that gets daughters of a particle from the MC truth information. Here is a quick recipe to find it.

- Go to the LXR main page.
  - Although the recipes on this page explicitly refer to Release 12, they will work for any release
- Click on the button that says "Start Browsing Stable 12 ATLAS Releases" - it should bring you to this page
- The default is **release_12_0_6** (near the top of the page) - should bring you here
- Next click on **general search** (near the bottom of the page) - should bring you to here
- In the field "Files Named" enter **.** and in the field "Containing" enter **getDaughters**
- You will get an output like here
  - You can see that all instances of getDaughters is in the BPhysExamples package
- **TIP:** - If you search for a method/variable that has a more generic name, you can get thousands of references.
  - For instance, if you typed **Muon** instead of getDaughters, the resulting output will give you 997 hits

6.6.3 Example 2

- If you want to see how to access various parameters for, say, an electron, you can do the following.
- In the field "Files Named" enter **.cxx** and in the field "Containing" enter **Electron.h**. This will give you a listing of all .cxx files where Electron.h is present.
- You can then go through these files to see how to access parameters of an electron object.

6.6.4 Example 3

- If you want to search a subset of files, e.g., `CBNT*.cxx`, then do the following:
  - In the `Files Named` field, enter `CBNT.*cxx`
  - Check the box `Advanced ((allow perl regex)`
  - Fill the usual in the field *Containing*

++ Example 4

- You can do an AND or OR with the following syntax:
  - If in the field *Containing*, you fill `a;b`, you will be search for `a AND b`
  - If you do `a,b`, then you will be doing `a OR b`

Complete: 🟨🟨🟨⬜
Responsible: Vivek Jain
Author: Vivek Jain 03 May 2007
Contributors: Vivek Jain
Last significantly modified by: Vivek Jain 03 May 2007
Not yet reviewed

Appendix A

Appendix

A.1 Basic Linux Commands

Linux has a hierarchical, unified filesystem (directories within directories; and files, directories, and device drivers are treated as files), supports 256-character filenames (avoid symbols and punctuation except for the dot (.) and note that you can have more than one nonadjacent dots in the filename - e.g. `this.is.okay`). All command line entries are case sensitive. Also note that Linux uses the slash (/) rather than the backslash in Windows (to separate directories. (e.g. `/home/user/top/nextdown/file.name`). There are three 'special' directories - . indicates the current directory, .. the directory above it and ∼ the users 'home' directory, the current directory when you log in.

You type commands inside a **shell** at the prompt (e.g >) and press ENTER. To find out what shell you are using type `echo $SHELL`. Most commands take optional arguments with minus signs e.g. `command -a - b -c` or `command -abc`. Type `command --help` to find out what the options are in most cases. The filename can contain wildcards such a * which means the command will be applied to all files matching the pattern. Be very careful when deleting files using wildcards!

The most commonly used Linux commands are:

Current Directory

`pwd` - To find out what your current directory is.

Change Directories

`cd dir1` - Change to directory `dir1` below the one you are in.
`cd /home/user/top/dir2` - Change to directory `dir2` from anywhere.
`cd ..` - Change to the directory above the one you are in.
`cd` - Change to your home directory.

Create Directory

`mkdir dir1` - Create directory `dir1` below the one you are in.
`mkdir /home/user/top/dir2` - Create directory `dir2` from anywhere.

Delete Directory

`rmdir dir1` - Delete directory `dir1` below the one you are in.
`rmdir /home/user/top/dir2` - Delete directory `dir2` from anywhere.
These only work if the directory is empty.

Copy Directory

`cp -R dir1 dir2` - Copy the contents of the directory `dir1`.
If directory `dir2` does not exist, it is created. Otherwise, it creates a directory named `dir1` within directory `dir2`.

List Directory

`ls` - list the files in the current directory.
`ls dir1` - List directory `dir1` below the one you are in.
`ls /home/user/top/dir2` - List directory `dir2` from anywhere.
Useful options are `-l` to produce a long listing, `-t` to sort by modification date and `-r` reverse the order e.g.
`ls -ltr` - Produce a long listing with the latest modification date last.

List File

`more file.txt` - List the contents of a (text) file on the screen.
Some people prefer to use `less` which is similar or `cat` which does the same but is much more basic.

Delete (Remove) File

`rm file1` - Delete file `file1` from the current directory.
`rm /home/user/top/file2` - Delete file `file2` from anywhere.
`rm *.txt` - Delete all the files ending in `.txt` in the current directory.
`rm -R dir1` - Recursively delete all files in all directories below `dir1` including `dir1` itself.

Copy File

`cp file1 file2` - Copy file `file1` into `file2`.
If `file2` does not exist, it is created; otherwise, `file2` is overwritten with the contents of `file1`.
`cp file1 dir1` - Copy file `file1` (into a file named `file1`) inside of directory `dir1`.

Rename (Move) File

`mv oldfile newfile` - Rename `oldfile` to `newfile` in the current directory.
`mv oldfile /home/user/top/oldfile` - Move `oldfile` to the directory `/home/user/top`.

Run Commands from a File

`source file` - Executes the commands stored in `file`. This would normally be a shell script `file.sh` or `file.csh`.

Complete: ▮▮▮▮▯
Responsible: Steve Lloyd
Author: Steve Lloyd 06 Oct 2005
Contributors: Steve Lloyd
Last significantly modified by: Steve Lloyd 06 Oct 2005
Not yet reviewed

A.2 Basic AFS Commands

AFS (the Andrew File System) is a world-wide distributed file system marketed years ago by a company called Transarc, later acquired by IBM who eventually released it to the public under the name OpenAFS. AFS permits easy sharing of files in a heterogeneous distributed environment (UNIXes, Windows) under a strong authentication service (Kerberos). The initially native AFS implementation of kerberos has meanwhile been replaced by a version of Kerberos 5 developed at KTH in Sweden (Heimdal).

For documentation see the OpenAFS Documentation User's Guide. For information about disk space see AFSdisk space in ATLAS (choose the "Table of contents" among the icons on top). There is also an (old) CERN AFS User's Guide and the SLAC AFS User's Guide.

A.2.1 AFS Commands

The following commands are useful for users. Another set is useful to system administrators. The bare minimum commands any user should know are: `klog` and `tokens`. AFS commands are at CERN Linux stored in `/usr/sue/bin`, and man pages in `/usr/bin/man`.

```
fs      – (FileSystem) commands to manage files and ACLs
klog    – obtain authentication token
knfs    – obtain authentication from non–AFS system (NFS) via translator
kpasswd – change authentication password
pts     – (ProTection Server) commands to manage ACL groups
rcp     – afs replacement for normal rcp
rsh     – afs replacement for normal rsh
tokens  – display all tokens
unlog   – discard all tokens
```

The `pts` command has a number of subcommands:

```
ad      adduser         add a user to a group
ap      apropos         search by help text
ch      chown           change ownership of a group
cg      creategroup     create a new group
cu      createuser      create a new user
del     delete          delete a user or group from database
e       examine         examine an entry
h       help            get help on commands
listm   listmax         list max id
listo   listowned       list owned groups
mem     membership      list membership of a user or group
rem     removeuser      remove a user from a group
ren     rename          rename user or group
setf    setfields       set fields for an entry
setm    setmax          set max id
```

The `fs` command has a number of subcommands:

```
ap      apropos         search by help text
checks  checkservers    check local cell's servers
checkv  checkvolumes    check volumeID/name mappings
cl      cleanacl        clean up access control list
co      copyacl         copy access control list
```

```
de       debug            set debugging info
df       diskfree         show server disk space usage
exa/lv   examine          display volume status
exp      exportafs        enable/disable translators to AFS
         flush            flush file from cache
flushv   flushvolume      flush all data in volume
getca    getcacheparms    get cache usage info
getce    getcellstatus    get cell status
gets/gp  getserverprefs   get file server ranks
h        help             get help on commands
la       listacl          list access control list
listc    listcells        list configured cells
lq       listquota        list volume quota
ls       lsmount          list mount point
me       messages         control Cache Manager messages
         mkmount          make mount point
         monitor          set cache monitor host address
         newcell          configure new cell
q        quota            show volume quota usage
rm       rmmount          remove mount point
sa       setacl           set access control list
setca    setcachesize     set cache size
setce    setcell          set cell status
sq       setquota         set volume quota
sets/sp  setserverprefs   set file server ranks
sv       setvol           set volume status
sy       sysname          get/set sysname (i.e. @sys) value
whe      whereis          list file's location
whi      whichcell        list file's cell
ws       wscell           list workstation's cell
```

A.2.2 Examples

The following examples are for the computing web pages:

```
> cd $COMPUTINGWEB
> fs help
> fs listacl
> pts membership atlas:www      - developers with permission to update the ATLAS
  web
> pts membership zp:software      - developers with permission to update the
  computing web pages
```

Originally taken from the ATLAS static web pages http://atlas-computing.web.cern.ch/atlas-computing/projects/externalLibsTools/afs.php and BaBar documentation http://www.slac.stanford.edu/BFROOT/www/Computing/Environment/NewUser/htmlbug/node50.html.

Complete:
Responsible: Steve Lloyd
Author: Traudl Kozanecki 20 Aug 2007
Contributors: Traudl Kozanecki, Steve Lloyd
Last significantly modified by: Steve Lloyd 28 Jan 2008
Not yet reviewed

A.3 Basic Python

Python is an *interpreted*, *interactive*, *object-oriented* programming language. It is increasingly used inside ATLAS as a scripting language. Athena Job Options files are written in Python. Full details with tutorials and documentation can be found at http://www.python.org. Python gets its name from the BBC series "Monty Python's Flying Circus".

You can execute Python as a script like other languages by putting

```
#! /usr/bin/env python
```

at the start or you can run the interpreter at the command line (use ctrl-d to quit).

```
> python
Python 2.3.4 (#1, Feb 18 2005, 12:15:38)
[GCC 3.4.3 20041212 (Red Hat 3.4.3-9.EL4)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 10
>>> print a
10
>>>
```

The unusual thing about Python compared with C, C++, Java, Perl etc is the lack of ";" at the end of statements and the lack of " " around blocks of code. Blocks of code are indicated by indentation. Comments start with # as in other languages.

Assignments

The main use of Python in ATLAS Job Options is to simply set variables:

```
EvtMax = -1
SkipEvents = 0
doHist = False
PoolESDOutput = "esd.pool.root"
```

You can also do multiple assignments:

```
x, y, z = 1, 2, 3
a = b = 123
```

and fill and extend arrays (called Lists):

```
PoolRDOInput = [ "g4digi1.pool.root", "g4digi2.pool.root"]
theApp.Dlls   += [ "AnalysisTools" ]
```

Control Structures

As stated above blocks of code are indicated by indentation. Remove the indentation at the end of the block e.g.:

```
if x < 5 or (x > 10 and x < 20):
        print "The value is OK."
```

```
if x < 5 or 10 < x < 20:
    print "The value is OK."
elif x >=20:
    print "The value is too big."
else:
    print "The value is in the middle."

for i in [1,2,3,4,5]:
    print "This is iteration number", i

names = ["John", "Paul", "Ringo", "George"]
for name in names:
    print name

x = 10
while x >= 0:
    print "x is still not negative."
    x = x - 1
```

`break` and `continue` allow you to exit a block prematurely or to skip the rest of the block and continue with the next iteration.

Complete: 🟥🟥⬜⬜⬜
Responsible: Steve Lloyd
Author: Steve Lloyd 04 Nov 2005
Contributors: Steve Lloyd
Last significantly modified by: Steve Lloyd 04 Nov 2005
Not yet reviewed

A.4 Glossary

A.4.1 Purpose of the ATLAS Glossary

- **Definitions of ATLAS and related acronyms**
- **Definitions (and acronyms) of ATLAS terminology and jargon**
- **Standard ways to write ATLAS terminology, to ensure consistency**
    - *If you decide that you must do things differently, e.g. capitalization, then at least do it in a consistent way*

A.4.2 Further sources

- Athena Startup Kit Glossary
- Grid Acronym Soup
- HEP Acronyms from FNAL

A.4.3 Main ATLAS Glossary

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

$ **AANT** Athena Nware NTuple.

$ **ABC** ATLAS Binary Chip (Inner Detector).

$ **ABCD** ATLAS Binary Chip in DMILL technology for SCT readout.

$ **ABCD3TA** ABCD version 3 with trim DACs, revised version.

$ **ACR** ATLAS Control Room.

$ **ADA** ATLAS Distributed Analysis system.

$ **ADC** Analogue to digital converter *or* ATLAS Distributed Computing.

$ **AFS** Andrew File System. A distributed network file system that enables file sharing form any machine running the AFS daemon.

$ **AGILe** A Generator Interface Library.

$ **AID** Analysis, Interpretation, Display (RTT).

$ **AIDA** Abstract Interface for Data Analysis.

$ **AIP** Alarm Integration Procedure.

$ **ALFA** Absolute Luminosity For ATLAS.

$ **Alpgen** Multi-parton process Monte Carlo event generator.

$ **AMI** See ATLAS Metadata Interface.

$ **AMT** Atlas Muon TDC.

$ **AOD** Analysis Object Data. Reduced size output of physics quantities from the reconstruction.

$ **API** Application Program Interface.

$ **ARA** AtlasProtected.AthenaROOTAccess

$ **ARC** Advanced Resource Connector - NorduGrid (Grid).

$ **ARDA** A Realization of Distributed Analysis of LHC (Grid).

$ **ARTEMIS** A research training network for experimental HEP and phenomenology.

$ **ASAP** Atlas Spectrometer Alignment Program.

$ **ASDBLR** Amplifier, Shaper, Discriminator, and Baseline Restorer. The analogue front-end chip used in the Transition Radiation Tracker.

$ **ASIC** Application-Specific Integrated Circuit. A custom-made chip.

$ **A-side** The two ends of ATLAS are called the 'A-side' and the 'C-side' ('B' is the central barrel). The A-side is along the positive z-axis and is in the direction of the airport and the Saleve.

$ **ASK** Athena Startup Kit.

$ **Athena** ATLAS offline software framework.

$ **ATCN** ATLAS point 1 Control Network.

$ **Atlantis** ATLAS standalone event display.

$ **ATLAS** A Toroidal LHC ApparatuS.

$ **ATLAS Metadata Interface (AMI)** Tool to find datasets and obtain detailed information on them.

$ **ATLCAL** Disk pool for ATLAS calibration data.

$ **ATLDATA** Disk pool for ATLAS tier-0 reconstructed data.

$ **ATLFAST** Software package for fast particle-level simulation.

$ **ATN** Atlas Testing Nightly. A testing framework FOR nightly builds of ATLAS software releases.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

$ **barrel** The central-rapidity region of the ATLAS detector.

$ **BC** See Bunch-crossing.

$ **BCID** See Bunch-Crossing Identification *and/or* Bunch-Crossing Identifier (ambiguous).

$ **BCM** Beam Conditions Monitor.

$ **BC-MUX** Bunch-crossing multiplexing. Used in the Level-1 Calorimeter Trigger to double the number of trigger towers per serial link.

$ **BCR** See Bunch Counter Reset.

$ **beam 1** The LHC beam which rotates clockwise when seen from above. In ATLAS it goes from positive z to negative z.

$ **beam 2** The LHC beam which rotates anti-clockwise when seen from above. In ATLAS it goes from negative z to positive z.

$ **BER** Bit-Error Rate.

$ **BDII** Berkely Database Information Index (LCGG GIIS replacement).

$ **BGA** Ball-grid array (a technology for connecting chips with very large numbers of connections).

$ **B-layer** The innermost layer of the Pixel Detector.

$ **BNL** Brookhaven National Lab.

$ **BOC** Back of Crate.

$ **BPM** Beam Position Monitor.

$ BPTX ATLAS beam pick-up detectors, 175 m from the detector.

$ BST Beam Synchronous Timing.

$ Bunch Counter Reset (BCR) Signal broadcast by the TTC system once per LHC orbit (88.924 microseconds) to control the phase of local bunch counters.

$ bunch-crossing (BC) Proton-proton bunch crossing in the the LHC. The bunch spacing is 24.95 ns.

$ bunch-crossing identification (BCID) The assignment of detector data to a specific bunch crossing.

(ROD_BCID) A 12-bit number that defines the bunch crossing at which an event occurred. It is provided by the TTC system to tag event fragments, and is reset each LHC orbit. Starts following LHC extractor gap, goes up to 3563.

$ ByteStream The raw data from the detector, consisting of hierarchically arranged fragments formatted in a subdetector-dependent way.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

$ C++ Primary programming language used in ATLAS.

$ CA Certification Authority (Grid).

$ CAF CERN Analysis Facility.

$ calorimeter cell The smallest unit of calorimeter information to be read out.

$ CANbus Control Area Network. A field bus for controlling and monitoring, used in the Detector Control System.

$ CASTOR CERN Advanced STORage Manager.

$ Cathode Strip Chamber (CSC) Muon chambers in the endcaps, used for both triggering and precision reconstruction.

$ CBNT ComBined N-Tuples.

$ CE Computing Element (Grid).

$ Central Trigger Processor The part of the Level-1 Trigger System that combines results from the Level-1 Calorimeter Trigger and Level-1 Muon Trigger to make the global yes/no Level-1 Trigger decision for each bunch crossing.

$ CERN European Laboratory for Particle Physics.

$ CINT A C/C++ interpreter that is embedded in ROOT.

$ CLHEP Class Library for HEP.

$ clock The 40.08 MHz clock linked to the LHC machine bunch-crossings.

$ Cluster Processor (CP) The part of the Level-1 Calorimeter Trigger that carries out the electron/photon and tau/hadron triggers.

$ CMM Common Merger Module of the Level-1 Calorimeter Trigger.

$ CM Coincidence Matrix of the Level-1 Muon Trigger.

$ CMT Configuration Management Tool, used for building software releases, package dependency management, and setup of the run-time environment.

$ CondDB See Conditions Database.

$ Conditions Database (CondDB) Contains records of the detector conditions for all data taking. This includes calibration and any other parameters required for the data analysis.

$ CondorG Grid Batch System.

$ ConfDB See Configuration Database.

$ Configuration Database (ConfDB) Stores the parameters necessary to describe the experiment's architecture, hardware and software components.

$ COOL LCGG Conditions Database Project.

$ coordinate system In ATLAS the x-axis points towards the centre of the LHC ring, the y-axis points upwards, and the z-axis points towards the airport and the Saleve, i.e. towards the A-side.

$ CORBA Common Object Request Broker Architecture.

$ COTS Commodity/Commercial Off-The-Shelf.

$ CP See Cluster Processor.

$ CPLD Complex Programmable Logic Device.

$ CPM Cluster Processor Module of the Level-1 Calorimeter Trigger.

$ CSC Computer System Commissioning *or* see Cathode Strip Chamber.

$ C-side The two ends of ATLAS are called the 'A-side' and the 'C-side' ('B' is the central barrel). The C-side is along the negative z-axis and is in the direction of the Jura.

$ CTB Combined Test Beam.

$ CTP See Central Trigger Processor.

$ CVS Concurrent Versioning System. Allows sharing of source code among a distributed development team. Code can be browsed and checked out. Records history of file modifications and allows retrieval of previous versions.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

$ **DAC** Digital to analogue converter.

$ **DAL** Data Access Library.

$ **DAQ** See Data Acquisition System.

$ **DAQ/HLT** Collective term for Data Acquisition System and High-Level Triggers.

$ **Data Acquisition System (DAQ)** System responsible for the assembly and permanent storage of events accepted by all three levels of the trigger system (Level-1, Level-2 and Event Filter). Comprises Data Flow, Online and Detector Control Systems.

$ **Data Collection (DC)** A subsystem of the Data Flow System, responsible for the movement of event data from the ROS to the Level-2 Trigger System, to the Event Filter, and also to mass storage.

$ **Data Collection Framework** A set of services used by all Level-2 and Event Builder applications, which provides a unified program structure and common interfaces to the Configuration Database, Run Control and other online software services.

$ **Data Flow Manager** Orchestrates the correct flow of data fragments between ROSs and SFIs. It is triggered by the Level-2 Supervisor, load balances the event-building tasks on the SFIs, and ensures that the ROSs do not overflow their internal memory buffers.

$ **Data Flow System (DF)** All software and hardware required for the management, transportation, and monitoring of physics data.

$ **DBMS** Database Management System.

$ **DC** Data Challenge *or* see Data Collection.

$ **dCache** Disk Cache System. Transparently manages the storage and exchange of data which is distributed across various storage devices.

$ **DCS** See Detector Control System.

$ **DDC** DAQ-DCS Communication.

$ **derandomizer** The memory in which data corresponding to a Level-1 Accept are stored before being read out.

$ **Detector Control System (DCS)** The system that monitors and controls physical parameters of the subsystems of the experiment, such as gas pressure, flow-rate, high voltage settings, low-voltage power supplies, temperatures, leakage currents, etc.

$ **DF** See Data Flow.

$ **DIAL** Distributed Interactive Analysis of Large datasets.

$ **DIG** Detector Interface Group.

$ **DM** Data Management.

$ **DN** Distinguished Name (Grid certificate).

$ **doublet** Part of the muon spectrometer, consisting of two layers of thin-gap chambers.

$ **DPD** Derived Physics Data.

$ DQ Don Quixote, distributed data manager (Grid).

$ DQM, DQMF Data Quality Monitoring, Data Quality Monitoring Framework.

$ DRD Derived Reconstruction Data.

$ DSB Doublet Slave Board of the Level-1 Muon Endcap Trigger.

$ DTMROC Digital Time Measurement Readout Controller. The digital front-end chip used in the Transition Radiation Tracker.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

$ EB See Event Builder, *or* ATLAS Executive Board *or* Tile Calorimeter Extended Barrel.

$ EBIF Event Builder Interface.

$ ECal Liquid argon Electromagnetic Calorimeter.

$ ECR See Event Counter Reset.

$ EDM Event Data Model.

$ EDG European Data Grid project.

$ EF See Event Filter.

$ EGEE Enabling Grid for E-sciencE.

$ ELMB Embedded Local Monitor Board of the Detector Control System.

$ EMB Liquid argon Electromagnetic Barrel calorimeter.

$ EMEC Liquid argon Electromagnetic Endcap Calorimeter.

$ EMI Electromagnetic interference.

$ EMS Event Monitoring Sampler.

$ endcaps The high-rapidity regions of the ATLAS detector.

$ ESD Event Summary Data. Provides sufficient information to re-run parts of the reconstruction, as AOD information may not be enough.

$ event The data resulting from a particular bunch-crossing.

$ Event Builder (EB) The subsystem that combines data corresponding to one event from all the sub-detectors. This takes place after acceptance by the Level-2 Trigger.

$ Event Counter Reset (ECR) Signal broadcast by the TTC system to reset local event counters.

$ **Event Filter (EF)** The third level of event selection, responsible for reducing the trigger rate to a value acceptable for permanent storage as well as doing data monitoring and calibration, using offline-style algorithms operating on complete events accepted by the Level-2 Trigger.

$ **event fragment** Generic term for a subset of event data. Examples are ROD and ROB fragments.

$ **Event_ID (EVID)** A number that identifies an event uniquely within a run.

$ **event type** See trigger type.

$ **EVO** Enabling Virtual Organizations.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

$ **FADC** Flash Analogue to Digital Converter.

$ **FC** File Catalogue.

$ **FCal** Liquid argon Forward Calorimeter.

$ **FE** See Front-end electronics.

$ **FE_BCID** A 12-bit number corresponding to the bunch number in the LHC machine, to identify the bunch crossing. It is generated locally in the RODs, reset by BCR, and used to cross-check against ROD_BCID when identifying event fragments to be read out.

$ **FE_L1ID** A number, of 3-4 bits, corresponding to the event number and generated locally in the RODs by counting Level-1 Accept signals. It is used to cross-check against L1ID when identifying event fragments to be read out.

$ **FIR** Finite-Impulse Response. A type of digital filter, used in Bunch-crossing identification.

$ **FLUKA** Monte Carlo program used to simulate electromagnetic and hadronic particle showers in the ATLAS detector. Used for radiation calculations.

$ **FORTRAN** Programming Language.

$ **FP420** Forward Proton tagging in the 420 metre region.

$ **FPGA** Field-Programmable Gate Array.

$ **front-end electronics (FE)** The detector subsystems which generate and send trigger data to the Level-1 Trigger, and event data to their RODs for transmission to the data acquisition system.

$ **FSI** Frequency Scanning Interferometry.

$ **FSM** Finite State Machine.

$ **FTP** File Transfer Protocol.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

$ GACL Grid Access Control List.

$ GANGA Gaudi/Athena Grid user interface.

$ Gaudi Data processing applications framework. Originally developed by and shared with LHCb experiment.

$ GCS Global Control Station of the Detector Control System.

$ GE Gigabit Ethernet.

$ GEANT Pan-European Gigabit Research and Education Network.

$ Geant 4 A general Monte Carlo simulation package for describing detector geometry and tracking particles through detector material. Used to simulate the response of the ATLAS detector.

$ GID Global event IDentifier.

$ GLIMOS Group Leader In Matters Of Safety.

$ gLite Lightweight middleware for Grid computing.

$ Globus Grid middleware.

$ GLUE Grid Laboratory Uniform Environment.

$ GNAM Gnam is Not AtlMon. Online monitoring software system.

$ GridFTP Protocol extensions to FTP for the Grid.

$ GSI Grid Security Infrastructure.

$ GUI Graphical User Interface.

$ GUID Globally Unique Identfier (Grid).

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

$ HBOOK Legacy histogramming package.

$ HCal Hadronic Calorimeter (Tile Calorimeter barrel, Liquid-argon Hadronic Endcap Calorimeter).

$ HEC Liquid-argon Hadronic Endcap Calorimeter.

$ HepMC C++ event record for Monte Carlo generators.

$ HEP High-energy physics.

$ HERWIG A Monte Carlo package for simulating Hadron Emission Reactions With Interfering Gluons.

$ High-Level Triggers (HLT) Collective term for the Level-2 Trigger and the Event Filter, the two trigger levels that are implemented primarily in software.

$ HLT See High-Level Triggers.

$ High-pT board Board of the Level-1 Muon Trigger system that implements the high-pT muon trigger.

$ HOLA High-speed Optical Link for ATLAS.

$ HSM Hierarchical Storage Management.

$ HTML HyperText Mark-up Language.

$ HVS Hierarchical Versioning System.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

$ ID See Inner Detector.

$ IDC Identifiable Container.

$ IGUI Integrated Graphical User Interface.

$ Inner Detector The inner tracking detector of ATLAS, made up of the Pixel, Semi-Conductor (SCT) and Transition Radiation (TRT) Trackers.

$ interval of validity (IOV) The time interval during which an entry in the Conditions Database is valid.

$ IOV See Interval Of Validity.

$ IP Interaction Point.

$ IPC Inter-Process Communication.

$ IS Information Service of the online software.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

$ Java Programming language.

$ JCOP Joint Controls Project.

$ JEM Jet/Energy Module of the Level-1 Calorimeter Trigger.

$ JEP See Jet/Energy-sum Processor.

$ jet element The smallest elements, 0.2x0.2 in eta-phi, used to form transverse-energy sums for the jet trigger. They are summed over the combined depth of the electromagnetic and hadronic calorimeters.

$ Jet/Energy-sum Processor (JEP) The part of the Level-1 Calorimeter Trigger that carries out jet, missing-ET and total-ET triggers.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

$ Kerberos A network authentication protocol.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

$ L1 See Level-1 Trigger System.

$ L1A See Level-1 Accept.

$ L1Calo See Level-1 Calorimeter Trigger.

$ L1ID A 24-bit number corresponding to the event number defined by counting Level-1 Accept trigger signals. It is provided by the TTC system to tag event fragments.

$ L1Muon See Level-1 Muon Trigger.

$ L2 See Level-2 Trigger System.

$ L2PU See Level-2 Processing Unit.

$ L2SV See Level-2 Supervisor.

$ LAr or Larg Liquid argon; see Liquid Argon Calorimeters.

$ LAN Local Area Network.

$ LCG LHC Computing Grid.

$ LCS Local Control Station of the Detector Control System.

$ LDAP Lightweight Directory Access Protocol.

$ Level-1 Accept (L1A) A signal generated by Central Trigger Processor when an event has met the Level-1 Trigger criteria. It is distributed by the TTC system.

$ Level-1 buffer Buffer (analogue or digital) in the front-end electronics that retains the event data until the Level-1 Accept result is received.

$ Level-1 Calorimeter Trigger (L1Calo) The part of the Level-1 Trigger System based on information from the calorimeters. Trigger objects are e.m. (electron/photon) showers, taus, jets, missing ET and total ET.

$ Level-1 Muon Trigger (L1Muon) The part of the Level-1 Trigger System based on information from the muon detectors. Trigger objects are high-pT muons.

$ Level-1 Trigger System (L1 or LVL1) The first level of event selection, consisting of the Level-1 Calorimeter and Muon Triggers and the Central Trigger Processor, responsible for reducing the event rate from the bunch-crossing rate of 40 MHz to no more than 75 kHz. Based on custom hardware and uses a subset of detector data. For accepted events, it issues Level-1 Accept to the front-end electronics.

$ Level-2 Processing Unit (L2PU) The application running on one of the Level-2 processors, from which the Level-2 Trigger decision is derived.

$ Level-2 Supervisor (L2SV) The interface between the Level-1 and Level-2 Triggers via the RoI Builder. It is responsible for distributing events to the Level-2 farm and manages the computing resources by means of load balancing algorithms. L2SV receives the final Level-2 Trigger decision on an event based on the result from the L2PUs.

$ Level-2 Trigger System (L2 or LVL2) The second level of event selection, responsible for reducing the trigger rate from about 75 kHz (upgradeable to 100 kHz) to a rate acceptable to the Event Filter, $\sim$2.5 kHz. Uses Regions-of-Interest from the Level-1 Trigger to selectively read out only certain parts of the detector.

$ LFN Logical File Name.

$ LHC Large Hadron Collider.

$ Liquid Argon Calorimeters (LAr or Larg) The barrel (EMB) and endcap (EMEC) electromagnetic calorimeters, the endcap hadronic calorimeters (HEC), and the forward calorimeters (FCal).

$ LMB Local control Monitor Board of the Level-1 Muon Endcap Trigger.

$ LSF Load Sharing Facility.

$ LS-Link Local slave link. A cable link between slave boards in the Level-1 Muon Endcap Trigger logic via which data are read out.

$ LTP Local Trigger Processor, a module used to general Level-1 Triggers when running independently of the Central Trigger Processor, e.g. for tests or calibration.

$ LUCID LUminosity measurement using Cerenkov Integrating Detector.

$ LUT Lookup Table.

$ LVDS Low-voltage Differential Signalling.

$ LVL1 See Level-1 Trigger System.

$ LVL2 See Level-2 Trigger System.

$ LVL3 See Event Filter.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

$ M&O Maintenance and Operation.

$ MBTS Minimum Bias Trigger Scintillator.

$ MC Monte Carlo simulation.

$ MDA Monitoring Data Archive.

$ MDS Monitoring and Directory Service (Grid).

$ MDT See Monitored Drift Tube.

$ Message Reporting System (MRS) A facility that allows software components of the TDAQ system to report error messages to other components.

$ MIBAK Backplane that connects modules of the MUCTPI.

$ MICTP Module of the MUCTPI that drives data to the CTP.

$ MIOCT Module of the MUCTPI that processes data for a muon spectrometer area of one octant in phi and half the detector in eta.

$ MIROD Module of the MUCTPI that supplies data to the Level-2 Trigger System (for RoI building) and to the Readout Buffers.

$ MONARC Models Of Networked Analysis at Regional Centres.

$ MoU Memorandum of Understanding.

$ Monitored Drift Tube (MDT) Muon chambers used for precision reconstruction, in both barrel and endcaps.

$ MOORE muon Object Oriented REconstruction.

$ MRS See Message Reporting System.

$ MSSM Minimal SuperSymetric Model.

$ MTBF Mean Time Between Failures.

$ MTTF Mean Time To Failure.

$ MUCTPI Level-1 Muon Trigger to CTP Interface.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

$ **NIM** Nuclear Instrumentation Module. A modular system of fast logic, used for trigger systems in particle-physics experiments in much simpler times (1960s-80s). Still in use, most commonly in test-beam triggering.

$ **NOVA** Networked Object-based Environment for Analysis.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

$ **OBK** Online Book Keeper of the online software.

$ **Octant** A collection of Thin Gap Chambers comprising one-eighth of a Level-1 Muon trigger plane in azimuth.

$ **OHP** Online Histogram Presenter.

$ **OHS** Online Histogramming Service.

$ **OKS** Object Kernel Support. A package that provides a simple, active, persistent in-memory object manager which is used to implement run-time configuration databases.

$ **OLE** Object Linking and Embedding.

$ **OMD** Operational Monitoring Display.

$ **OO** Object Oriented software.

$ **Open Science Grid** US Grid system.

$ **Orbit** A signal transmitted by the LHC to the TTC at a fixed point in the LHC cycle. The ORBIT signal is the broadcast to the TTC partitions.

$ **OS** Operating System.

$ **OSG** See Open Science Grid.

$ **OTSMOU** Operation Task Sharing and Maintenance & Operation Update.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

$ Pacman Package Manager for software.

$ partition The physical or logical separation of one or more elements of the experiment into mutually exclusive subsets. Allows subdetectors to work independently; see TDAQ partition.

$ PASTA LCGG technology tracking team.

$ PAW Physics Analysis Workstation (legacy data analysis package).

$ PBS Portable Batch System.

$ PESA Physics and Event Selection Architecture.

$ PID Partition Identifier.

$ pivot plane Plane of chambers that defines the RoI position in the muon Resistive Plate Chambers or Thin Gap Chambers. Equivalent to 'reference plane'.

$ Pixel Tracker The innermost layers of the Inner Detector.

$ PMG See Process Manager.

$ PMT Photomultiplier tube.

$ POOL Pool Of persistent Objects for LHC.

$ PPM PreProcessor Module of the Level-1 Calorimeter Trigger.

$ PPr See PreProcessor.

$ PreProcessor (PPr) The part of the Level-1 Calorimeter Trigger that digitizes the calorimeter signals, does bunch-crossing identification, and uses a lookup table to do final ET calibration.

$ PPrASIC PreProcessor ASIC of the Level-1 Calorimeter Trigger.

$ PPrMCM PreProcessor Multi-Chip Module of the Level-1 Calorimeter Trigger.

$ prescale factor Reduces the rate of events accepted by a specific Level-1 Trigger logic item.

$ Process Manager (PMG) Performs basic job control of TDAQ software (starting, stopping, and monitoring basic status).

$ processing node The hardware on which one or more Level-2 Processing Units or Event Filter processing tasks run.

$ PROOF Parallel ROOT Facility.

$ PS-Pack Patch-panel and slave-board package of the Level-1 Endcap Muon Trigger.

$ PV Primary Vertex.

$ PVSS Prozessvisualisierungs-und Steuerungs-System (Process Visualization and Control System). Commercial software used by the Detector Control System.

$ PYTHIA A Monte Carlo program used to generate simulated proton-proton interactions for various physics processes.

$ Python An interpreted, interactive, object-oriented, open-source programming language.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

$ RAL Relational Access Layer *or* Rutherford Appleton Laboratory.

$ raw data Data provided by the front-end electronics to the readout buffer.

$ RB Resource Broker (Grid).

$ RC See Run Control, *or* Replica Catalog (Grid).

$ RCD ROD crate DAQ.

$ RDBMS Relational Database Management System.

$ RDO Raw Data Object. The data as it comes off the detector.

$ readout buffer (ROB or ROBin) A standard module that receives data from one or more RODs via standard Readout Links, passes on request a subset of the data to the Level-2 Trigger, and buffers the data until a Level-2 Trigger decision has been reached. It then sends the data to the Event Builder.

$ Readout Driver (ROD) The detector-specific last element in the readout chain that is still considered part of the front-end electronics. Collects data streams from the Derandomizers and merges them into a single stream which is fed via a standard Readout Link into a Readout Buffer.

$ Readout Link (ROL) The ATLAS-standard data-transmission link between a ROD and a ROB.

$ ROOT A class library for data analysis, with extensive facilities including data display, persistency, minimization, etc.

$ Readout System (ROS) The first element in the readout chain that is considered part of the Data Acquisition System. Collects data from Readout Drivers via Readout Links and supplies it to the Level-2 Trigger and the Event Builder.

$ Receiver Station Modules into which analogue trigger-tower signals from the calorimeters are received, and their gains adjusted to be on a calibrated scale proportional to ET. Signals are also available for waveform monitoring by the calorimeter groups.

$ RecExCommon A CMT run-time environment for reconstruction.

$ reference plane Plane of chambers that defines the RoI position in the muon Resistive Plate Chambers or Thin Gap Chambers. Equivalent to 'Pivot plane'.

$ region-of-interest (RoI) A geographical region of the experiment, identified by the Level-1 Trigger System as containing candidates for Level-2 Trigger objects requiring further computation.

$ Region-of-Interest Builder (RoIB) A unit that collects and formats level-1 Region-of-Interest information and sends it to the Level-2 Supervisor for use by the Level-2 Trigger.

$ Resistive Plate Chamber (RPC) Muon chamber used for the Level-1 Muon Trigger in the barrel region.

$ RFIO Remote File I/O.

$ RHEL Redhat Enterprise Linux.

$ RIO Reconstruction Input Object.

$ RLS Replica Location Service (Grid).

$ ROB or ROBin See Readout Buffer.

$ ROB fragment Set of ROD fragments for one event within one Readout Buffer.

$ ROC Readout Crate (specific implementation of a ROS) *or* ROD Controller module for Liquid Argon Calorimeters.

$ ROD See Readout Driver.

$ ROD_BCID See Bunch-Crossing_Identifier.

$ ROD_BUSY A signal to indicate that the ROD is busy, used to inhibit Level-1 Triggers.

$ ROD fragment Data provided by the front-end electronics to the Readout System for one event.

$ RoI See Region-of-Interest.

$ RoIB See Region-of-Interest Builder.

$ ROL See Readout Link.

$ ROOT A class library for data analysis.

$ ROS See Readout System.

$ ROS fragment The set of ROB fragments for one event within a ROS. Deprecated.

$ RPC See Resistive Plate Chamber.

$ RSL Resource Specification Language (Grid).

$ RT Real-time.

$ RTAG Requirements Technical Assessment Group (LCGG).

$ Run Control The online software system that controls data-taking by coordinating operation of the TDAQ subsystems, online software components, and external systems.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

$ SBC Single-board Computer. For example, controlling the VMEbus and running software for the modules in the VME crate.

$ SCADA Supervisory Control and DAQ.

$ **SCT** Semiconductor Tracking detector.

$ **SCX** Surface Control Room.

$ **SDP** Software Development Process.

$ **SDO** Simulation Data Objects.

$ **SDX** Surface Counting Room.

$ **SE** Storage Element (Grid).

$ **SEAL** The Shared Environment for Applications at LHC.

$ **SFI** See Sub-Farm Input.

$ **SFO** See Sub-Farm Output.

$ **SG** See Storegate.

$ **SGE** Sun Grid Engine.

$ **SI2K** SpecInt2000 CPU benchmark.

$ **SIM** Simulated Data, in RAW Format.

$ **SIT** Software Infrastructure Team of ATLAS.

$ **SL** Scientific Linux *or* Sector Logic of the Level-1 Muon Trigger.

$ **SLC** Scientific Linux CERN.

$ **SLHC** Super LHC. Refers to LHC after major future luminosity upgrade.

$ **SLIMOS** Shift Leader In Matters Of Safety.

$ **S-Link** Simple Link interface. Protocol for data transmission on Readout Links.

$ **SRM** Storage Resource Manager (Grid).

$ **STL** Standard Template Library of C++.

$ **StoreGate** The transient data store for ATHENA.

$ **Sub-Farm Input (SFI)** The part of the Data Collection subsystem where full events are built by the Event Builder.

$ **Sub-Farm Output (SFO)** The part of the Data Collection subsystem where complete events received from the Event Filter are output to mass storage.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

$ **TAG** Event-level metadata.

$ **TCP/IP** Transmission Control Protocol/Internet Protocol.

$ **TDAQ** Collective term for Trigger, Data Acquisition and Detector Control systems.

$ **TDR** Technical Design Report.

$ **TDS** Transient Data Store.

$ **TES** Transient Event Store.

$ **TGC** See Thin Gap Chamber.

$ **Thin Gap Chamber (TGC)** Muon chamber used for the Level-1 Muon Trigger and precision muon tracking in the endcap regions.

$ **TileCal** See Tile Calorimeter.

$ **Tile Calorimeter (TileCal)** Hadronic barrel and extended-barrel calorimeters of ATLAS, using scintillating tiles as active medium.

$ **Timing, Trigger and Control (TTC)** The system that provides and distributes trigger signals (e.g. Level-1 Accept), timing signals (e.g. LHC clock), and fast control signals to the various subsystems of ATLAS.

$ **TLA** Three-letter acronym.

$ **TOF** Time Of Flight.

$ **trigger chamber** Generic term for Resistive Plate Chambers and Thin Gap Chambers, used in the Level-1 Muon Trigger.

$ **trigger menus** The set of trigger conditions in use at any particular time. They specify a list of items, each with its ET threshold and multiplicity, and the logic to be applied to them.

$ **trigger tower** The smallest element of calorimeter information used in the Level-1 Calorimeter Trigger, with dimensions of approximately 0.1x0.1 in eta-phi and summed over the full depth of either the electromagnetic or hadronic calorimeter.

$ **trigger type** An 8-bit word transmitted with the Level-1 Accept and giving information about the type of event, e.g. physics or calibration.

$ **TriP** Trigger Presenter online software.

$ **Triplet** Three layers of Thing Gap Chambers.

$ **TRT** Transition Radiation Tracker of the Inner Detector.

$ **TSB** Triplet slave board of the Level-1 Muon Endcap Trigger.

$ **TTC** See Timing, Trigger and Control.

$ **TTCex** TTC encoder/transmitter.

$ **TTCmi** TTC machine interface.

$ TTCrx TTC receiver chip. An ASIC that delivers the decoded and de-skewed TTC signals, bunch and event counters required by front-end electronics controllers.

$ TTCvi TTC VME interface module. Delivers the TTC A-channel and B-channel signals to the TTC transmitter crate. The A-channel transmits only the Level-1 Accept signal. The B-channel transmits framed and formatted commands and data.

$ TURL Transfer URL (Grid).

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

$ U Unit of length used to measure rack height, equal to 1.75 inches (4.45 cm).

$ UI User Interface.

$ UML Unified Modelling Language.

$ URL Universal Resource Locator (address used by worldwide web).

$ US15 ATLAS underground service area. On the positive-x side of UX15, i.e. inside the LHC ring.

$ USA15 Main ATLAS underground electronics cavern. On the negative-x side of UX15, i.e. outside the LHC ring.

$ UX15 ATLAS underground experimental cavern.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

$ VCSEL Vertical-Cavity Surface-Emitting Laser.

$ VHDL VHSIC (Very high speed integrated circuit) Hardware Description Language. A language for specifying the designs of electronic systems, widely used for FPGA firmware.

$ VMEbus Versa-Module Euro. A crate backplane bus system.

$ VO Virtual Organization (Grid).

$ VOMS Virtual Organization Membership Service (Grid).

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

$ **WAN** Wide Area Network.

$ **WLS** Wavelength-Shifting.

$ **WMS** Workload Management System (Grid).

$ **WN** Worker Node (Grid).

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

$ **XML** Extensible Markup Language.

$ **ZDC** Zero-Degree Calorimeter.

$ **ZEBRA** Legacy data management system.

Originally taken from http://wlav.home.cern.ch/wlav/athena/athask/glossary.html and http://atlas-proj-computing-tdr.web.cern.ch/atlas-proj-computing-tdr/Html/Computing-TDR-72.htm#pgfId-1011143

Workbook Glossary

Merged Computing Workbook glossary, Style Guide glossary, and some others

Complete: 🟩
Responsible: Eric Eisenhandler
Author: Wim Lavrijsen 27 May 2005
Contributors: Eric Eisenhandler, Wim Lavrijsen, Steve Lloyd
Last significantly modified by: Eric Eisenhandler 22 Aug 2008
Not yet reviewed

–