Table of Contents

Table of Contents

WISComputingguide

Atlas Terms on Babylon

Add this glossary to Babylon. (06 Dec 2007)

Running on lxplus

Login

You have to get an account from Cern. Instructions to do it are at:
https://twiki.cern.ch/twiki/bin/view/Atlas/WorkBookGetAccount

After getting an account you have to connect to lxplus.cern.ch (the public computer of Cern) using SSH program. A very convenient program is PuTTY which can be downloaded freely from: http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe (There is no installation, just double click on the file after downloading it).

Really not necessary: To make the unix environment more friendly I use the files

- .alias
- .zshrc

Just put them in your home directory, and they will be available on your next login. If you have your own aliases, shortcuts or configuration files that make your life easy feel free to update these files

Setting the environment

This section follows: https://twiki.cern.ch/twiki/bin/view/Atlas/WorkBookSetAccount

Setting your environment for the 1st time

The environment of lxplus is based on CMT, which is a Configuration Management Environment. The CMT is responsible (the way I understand it) to manage the different packages in Athena (For more details have a look at: http://www.cmtsite.org/). Anyway, no matter what the CMT is, you have to configure it. For this task you have to create the CMT home directory, and prepare a working directory.

```
cd scratch0/
mkdir cmthome
mkdir testarea
```

Now you have to decide the version of Athena you are interested to run, for example 12.0.5, 12.0.4 or 11.0.5. In what follows I describe how to run version 12.0.5, but feel free (and cross your fingers while doing so) to change it to your desired version. Here we create two work directories, one for version 12.0.5 and one for 11.0.5.

```
mkdir testarea/12.0.5
mkdir testarea/11.0.5
```

To setup the CMT:

```
cd cmthome
source /afs/cern.ch/sw/contrib/CMT/v1r19/mgr/setup.sh
```

The last line tells the shell to execute the script setup.sh. From time to time this file is updated, so choose the updated one. Be careful this will not work properly if you have a directory ~/cmt. Now you have to put a requirements file. You can write it in a text editor or just put: requirements

The macro ATLAS_TEST_AREA points to your working area. If you want to change your working area you have to change this macro to point to the right directory.

To configure the CMT according to the requirements file just run:

```
cmt config
```

Configure the environment of your shell must do every login

Every time you login (in every shell) you have to choose the version of Athena. Here we choose to work with 12.0.5.

```
cd ~/scratch0/cmthome
source ~/scratch0/cmthome/setup.sh -tag=12.0.5
```

If you choose to work with version 11.0.5 you have to run:

```
cd ~/scratch0/cmthome
source ~/scratch0/cmthome/setup.sh -tag=11.0.5
source /afs/cern.ch/atlas/software/dist/11.0.5/Control/AthenaRunTime/*/cmt/setup.sh
```

To check that it works, type:

```
cmt show path
```

You should see all the paths of the CMT packages, among them you should see the path to your working area.

Job Options in Athena

Athena is a framework for all the packages in Atlas. The way to choose what to execute is by writing what you want in a Python script, this is the "jobOption.py" file. The easiest way to run something on Athena is to locate the appropriate job option file and edit the parameters to match your needs. if you are interested to know more have a look at: http://wlav.web.cern.ch/wlav/joboptions/ (I didn't bother to look at it). The way to run something is using the command:

```
athena jobOptions.py
```

This will execute Athena according to the job option file. While executing Athena sends a log to the standard output. It is recommended to save the log, because sometimes it has errors and you can view them after the execution. You can save it using: :

```
athena jobOptions.py |& tee logfile.log
```

I like to paint the lines according to their status using the file .color.awk that I placed in my home directory, and I run:

```
athena jobOptions.py |& tee logfile.log | color
```

Running Hello World

It is always fun to run a Hello World program because it gives you the feeling that you are on the right track. So we follow here the instruction of the Atlas WokBook:
https://twiki.cern.ch/twiki/bin/view/Atlas/WorkBookRunAthenaHelloWorld

Execute:

```
cd ~/scratch0/testarea/12.0.5
cmt co -r UserAnalysis-00-09-10 PhysicsAnalysis/AnalysisCommon/UserAnalysis
```

This will check-out the relevant packages. Checking out is to take a package for a local use and edit (I think…). For version 11.0.5 replace UserAnalysis-00-09-10 with UserAnalysis-00-05-11.

Go to the relevant directory, setup and compile the package:

```
cd PhysicsAnalysis/AnalysisCommon/UserAnalysis/cmt/
source setup.sh
gmake
```

Finally, go to the run directory, get the job option file and run Athena:

```
cd ../run
get_files HelloWorldOptions.py
athena HelloWorldOptions.py |& tee helloWorld.log | color
```

The get_files command looks for a job options file with the name indicated and copy it to your current directory. If you see the following lines (disregard the FATAL error lines) it means that you ran it successfully.

```
…
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
HistogramPersis...   INFO "CnvServices": ["HbookHistSvc", "RootHistSvc"]
```

Running Batch Jobs

Every process that takes more than an hour (more or less) is killed by lxplus. Instead there are queues for jobs where you can place your own job. First you have to write a script to run your job. Here we show how to run HelloWorld. Create a script file, for example 'myjob.sh' with a text editor and put inside (but only after you checked out the package built it and got the job option file):

```
#!/bin/bash
cd ~/scratch0/cmthome
source ~/scratch0/cmthome/setup.sh -tag=12.0.5
cd ~/scratch0/testarea/12.0.5/PhysicsAnalysis/AnalysisCommon/UserAnalysis/cmt
source setup.sh
cd ~/scratch0/testarea/12.0.5/PhysicsAnalysis/AnalysisCommon/UserAnalysis/run
athena.py HelloWorldOptions.py
```

The first line opens a new shell for the job. The 2nd-3rd lines configure Athena for version 12.0.5. The 4th-last lines run Athena. You have to change the permissions of your script to be executable file:

```
chmod +x myjob.sh
```

And then place the job in the queue:

```
bsub -q 8nm myjob.sh
```

This will place your job in a queue for processes that run under 8 minutes. For processes that take up to 1 hour you have to replace 8nm with 1nh, for 8 hours 8nh, one day 1nd, one week 1nw.

There are a lot more options for this. You can check the status of your job by:

```
bjobs
bjobs -l jobID
```

and you can kill a job:

```
bkill jobID
```

You will be notified through email when the job is over, and where to find the log file.

ATLFast to CBNT (Combined NTuple)

Following: http://eduard.home.cern.ch/eduard/HowTo.htm

If you haven't run Hello World before then you must(???) execute:

```
cd ~/scratch0/testarea/12.0.5
cmt co -r UserAnalysis-00-08-43 PhysicsAnalysis/AnalysisCommon/UserAnalysis
cd PhysicsAnalysis/AnalysisCommon/UserAnalysis/cmt/
source setup.csh
gmake
```

After doing so get the following jobOptions

```
cd ~/scratch0/testarea/12.0.5/PhysicsAnalysis/AnalysisCommon/UserAnalysis/run
get_files POOLtoAtlfasttoCBNT.py Atlfast_MakeROOTFile.py Atlfast_ReadPOOLFile.py
chmod u+w POOLtoAtlfasttoCBNT.py Atlfast_MakeROOTFile.py Atlfast_ReadPOOLFile.py
```

Edit these files to read the desired input pool files, and write the desired ntuple output files.

Also chnge the include files to point to your edited jopOptions files. For example change:

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

to

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

Run Athena:

```
athena POOLtoAtlfasttoCBNT.py |& tee atlfast.log | color
```

You should get a CBNT NTuple file as output.

To add truth information to your NTuple you can include the file 'CBNT_Truth_jobOptions.py' (Thanks for Markus Warsinsiki for that) in the POOLtoAtlfasttoCBNT.py jobOptions file.

ATLFast to CBNTAAN (CBNT Athena Aware)

I don't really undersatnd the difference from the previous CBNT, and I'm not sure which one you should use.

Create the following jobOptions file and run it in athena instead of POOLtoAtlfasttoCBNT.py.

```
##############################################################################
#==================================
# Number of events and OutputLevel
#==================================
if not 'EvtMax' in dir():
```

```
  EvtMax = 99000
theApp.EvtMax = EvtMax
MessageSvc = Service( "MessageSvc" )
MessageSvc.OutputLevel = WARNING
MessageSvc.defaultLimit = 9999999

#include these for events in .pool.root file
include( "AthenaPoolCnvSvc/ReadAthenaPool_jobOptions.py" )
include( "EventAthenaPool/EventAthenaPool_joboptions.py" )

# load relevant libraries
include( "PartPropSvc/PartPropSvc.py" )
theApp.Dlls += [ "GaudiAlg" ]
theApp.Dlls += [ "AtlfastAlgs" ]
theApp.ExtSvc += [ "AtRndmGenSvc" ]
theApp.Dlls   += [  "GeneratorObjectsAthenaPoolPoolCnv" ]

EventSelector = Service( "EventSelector" )
EventSelector.InputCollections = [ "McEvent.root" ]

# set up the Atlfast sequence and run Atlfast
include( "AtlfastAlgs/Atlfast_ConfigAlgs_NoFastShower.py" )

# make standard atlfast AAN
include("AtlfastAlgs/Atlfast_MakeAAN.py")
AANTupleStream.OutputName = "atlfast.aan.root"
##############################################################################
```

AOD Analysis

Following: https://twiki.cern.ch/twiki/bin/view/Atlas/WorkBookStartingAODAnalysis
https://twiki.cern.ch/twiki/bin/view/Atlas/TriesteAnalysisTutorial1203 Also using Arie's document:
REMEMBER TO ADD THE FILE LATER

Preparations

Go to the test area and check out the UserAnalysis package,and go to the cmt directory:

```
cmt co -r UserAnalysis-00-08-43 PhysicsAnalysis/AnalysisCommon/UserAnalysis
cd PhysicsAnalysis/AnalysisCommon/UserAnalysis/cmt
```

Change the requirements file in that directory to be:

```
package UserAnalysis

author Ketevi A. Assamagan
```

After changing the requirements file setup the environment.

```
cmt config
source setup.sh
gmake
```

Get the necessary files:

```
cd ../run
get_files AnalysisSkeleton_topOptions.py
```

Change the field *EventSelector.InputCollections* to the input AOD file.

EventView / SusyView /

Following: https://twiki.cern.ch/twiki/bin/view/Atlas/SusyView The SusyView works only(???) with version 11.0.5.

IMPORTANT!! Instead of setting the CMT environment as described in 2.2.2 run the following line:

```
cd ~/scratch0/cmthome
source setup.sh -tag=11.0.5,groupArea
```

Check-out the SusyView package, setup a makefile and make it:

```
cd ~/scratch0/testarea/11.0.5
cmt co -r SUSYView-00-00-09 PhysicsAnalysis/SUSYPhys/SUSYView
cd PhysicsAnalysis/SUSYPhys/SUSYView/SUSYView-*/cmt
source setup.sh
cmt bro cmt config
cmt bro gmake
cd ../run
```

SusyView on ATLFast

The job options file for ATLFast is different (how???) from that of the full simulation.

```
get_files SUSYViewAtlFast_topOptions.py
```

At the beginning of the job options file add the following lins:

```
mode='test'
file='test.pool.root'
theApp.EvtMax=10
```

The EvtMax indicates the number of events to process. The file variable points to the ATLFast AOD file. The mode variable is just a prefix to the output file names of the SusyView ntuple files, you can change it to whatever name you like.

There are two output files: Reco and Truth.

- Reco – Has inside the reconstructed particles, and the Truth particles that match to the Reco particles.
- Truth – The Truth particles, including neutrinos but without the LSP.

Both files have the structure of EV ntuple files, with the different views inside.

Run Full Simulation

Following: https://twiki.cern.ch/twiki/bin/view/Atlas/WorkBookSimulation.

Generating Events

See Section 2.6.

Simulation

Go to the run directory.

```
cd ~/scratch0/testarea/12.0.5/PhysicsAnalysis/AnalysisCommon/UserAnalysis/run
```

Get the Geant4 job options file, and the PDG table.

```
get_files G4Atlas_Sim.py
get_files PDGTABLE.MeV
```

Then create a job options file ' mySimOptions.py' that has the following parameters:

```
#--- Detector flags -----------------------------------------
from AthenaCommon.DetFlags import DetFlags
# - Select detectors
DetFlags.ID_setOn()
DetFlags.Calo_setOn()
DetFlags.Muon_setOn()
#
DetFlags.simulate.Truth_setOn()

#--- Simulation flags ---------------------------------------
from G4AtlasApps.SimFlags import SimFlags
SimFlags.import_Flags('atlas_flags')

SimFlags.SimLayout.set_Value('ATLAS-CSC-01-02-00')
SimFlags.PersistencyHit.set_Value("g4hits.pool.root")

SimFlags.KinematicsMode.set_Value('ReadGeneratedEvents')
SimFlags.EvgenInput.set_Value("pythia.pool.root")

theApp.EvtMax = 10

#---  Output printout level ----------------------------------
#output threshold (2=DEBUG, 3=INFO, 4=WARNING, 5=ERROR, 6=FATAL)
#you can override this for individual modules if necessary
MessageSvc = Service( "MessageSvc" )
MessageSvc.OutputLevel = 3
```

Change the EvtMax to the maximum number of events to simulate. **Don't set it to -1 since it will be overridden to 3 by the G4Atlas_Sim.py.** Change the EvgenInput to the input file. Change the SimLayout to the layout of the detector. Change the PersistencyHit to the output file name (**I think…**)

Run Athena:

```
athena mySimOptions.py G4Atlas_Sim.py |& tee sim.log | color
```

Digitization

Create the job option file ' myDigiOptions.py' that has the following parameters:

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

PoolHitsInput = ["g4hits.pool.root"]
PoolRDOOutput = "g4digi.pool.root"

# Run through EvtMax events; if -1 run to end of file

EvtMax = -1

# Detector description
DetDescrVersion='ATLAS-CSC-01-02-00'

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

Change the DetDescrVersion to the layout of the detector. Change PoolHitsInput to the input file. Change PoolRDOOutput to the output file.

Run Athena:

```
athena myDigiOptions.py Digitization/AtlasDigitization.py |& tee digi.log | color
```

Reconstruction

Get a general job options file:

```
get_files RecExCommon_topOptions.py
```

Create your own job options file ' myRecOptions.py'

```
# For ESD Production
doWriteESD = True
doWriteAOD = True

# Detector description
DetDescrVersion='ATLAS-CSC-01-02-00'

# doTrigger = False # Do not run trigger simulation as it breaks in 12.0.1

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

# suppress the production of ntuple and histogram files
doCBNT = False
doHist = False

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

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

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

Change DetDescrVersion to the detector layout. Change PoolRDOInput to the input file. Change the flags doWriteESD, doWriteAOD to set the desired output format. Change PoolESDOutpt/PoolAODOutpt to the output file name.

Run Athena:

```
athena myRecOptions.py RecExCommon_topOptions.py |& tee rec.log | color
```

Running locally on physdsk

This instruction is relevant only for version 12.0.5 (See remark at 3.2.1). The instructions are intended for tcsh users. If you use bash or zsh change the name of the script from 'script.csh' to 'script.sh', and instead of using 'setenv' use 'export'.

Login

Log in to physdsk1 using SSH in a similar way to what was described in 2.1.

Setting the environment on physdsk

We do it analogically to the way it is done on lxplus (see 2.2) The location of the Atlas software is in: /opt/exp_soft/atlas/prod.

Setting your environment for the 1st time

First configure the CMT, and make it work with SSH authorization.

```
cd ~
mkdir cmthome
mkdir testarea
mkdir testarea/12.0.5
cd cmthome
source /opt/exp_soft/atlas/prod/releases/rel_12-2_1/CMT/v1r19/mgr/setup.csh
```

The last line indicates which kit version to handle. Different versions located in different kits. The version 12.0.5 is located in rel_12-2_1. Put the following requirements file in the cmthome directory:

```
set CMTSITE STANDALONE
set SITEROOT /opt/exp_soft/atlas/prod/releases/rel_12-2_1

macro ATLAS_DIST_AREA ${SITEROOT}

macro ATLAS_TEST_AREA ${HOME}/testarea

apply_tag projectArea
macro SITE_PROJECT_AREA ${SITEROOT}
macro EXTERNAL_PROJECT_AREA ${SITEROOT}

apply_tag setup
apply_tag simpleTest

use AtlasLogin AtlasLogin-* $(ATLAS_DIST_AREA)

set CMTCONFIG i686-slc3-gcc323-opt

set DBRELEASE_INSTALLED  2.6
```

Create the setup scripts:

```
cmt config
```

To create SSH key (Taken from http://cvs.web.cern.ch/cvs/howto.php#accessing): Go to the directory '~/.ssh', if it doesn't exist create it, and create SSH key there.

```
mkdir ~/.ssh
ssh-keygen -t rsa1
```

It is important to create it using RSA protocol 1 (-t rsa1). You should get interactive questions that look like that (See instruction what to do after the blob):

```
Generating public/private rsa1 key pair.
Enter file in which to save the key (/srv01/agrp/ohads/.ssh/identity):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /srv01/agrp/ohads/.ssh/identity.
Your public key has been saved in /srv01/agrp/ohads/.ssh/identity.pub.
The key fingerprint is:
80:27:2a:25:63:3f:00:a6:95:ec:d8:d6:62:af:d6:2e ohads@physdsk1
```

- The first question (2nd line) ask for the file name and location of the key. Just press enter to take the default.
- The next two questions (3rd and 4th lines) ask for a pass phrase. Enter a pass phrase of your choosing.
- The following lines indicate the creation of the identity files (their name is the name you gave at your first question).

After you create the key you have to copy it to your account at lxplus and make it be recognized by the SSH server.

```
scp ~/.ssh/identity.pub USERNAME@lxplus.cern.ch:~
```

USERNAME is your username on lxplus.

Log in to your account on lxplus and do the following:

```
/afs/cern.ch/project/cvs/dist/bin/set_ssh
cat ~/identity.pub >> ~/.ssh/authorized_keys
chmod 755 ~/public ~/.ssh ~/.ssh/authorized_keys
```

You can logout of lxplus.

Create the file '~/.ssh/config' with a text editor and put inside:

```
Host lxplus.cern.ch lxplus
Protocol 2
PubkeyAuthentication no
PasswordAuthentication yes

Host isscvs.cern.ch isscvs
Protocol 1
ForwardX11 no
IdentityFile ~/.ssh/identity

Host atlas-sw.cern.ch atlas-sw
Protocol 1
ForwardX11 no
IdentityFile ~/.ssh/identity
```

Configure the environment of your shell - must do every login

Run the setup scripts, can be done only for version 12.0.5 (See remark in 3.2.1).

```
source ~/cmthome/setup.csh -tag=12.0.5
```

To check if it works go to the test area and show the path:

```
cd ~/testarea/12.0.5
cmt show path
```

You should see the path to all the projects.

Now you have to set the CVS client to use ssh authorization: Taken from http://cvs.web.cern.ch/cvs/howto.php#accessing

```
setenv CVSROOT :ext:USERNAME@atlas-sw.cern.ch:/atlascvs
setenv CVS_RSH ssh
```

USERNAME is your username on lxplus.

Make the SSH agent use the identity file you created in 3.2.1.

```
eval `ssh-agent`
ssh-add ~/.ssh/identity
ssh-add -l
```

The second line should ask for the pass phrase you used when you created the key (See 3.2.1) The last line should show the RSA key you've added to the SSH agent.

Running Athena

You run Athena locally in the same way you run it on lxplus.

Running Batch jobs on physdsk1

Create a script file, for example 'myjob.sh'. Type inside:

```
#!/bin/sh
#
# As usual, the first line defines the shell
# Lines beginning '#BSUB' are LSF directives
#
# Give the job a name
#
#PBS -N pythiaS
# ----------------------------------------
#   How many need-nodes and parallel proceses
#PBS -l nodes=1:ppn=1
# ----------------------------------------
#
# We don't need much time so the X queue is fine
#
#PBS -q S
#
# The output will be in pythia.log
#
cd $PBS_O_WORKDIR
#

set -v
cd ~/scratch1/cmthome
source setup.sh -tag=12.0.5
cd ~/testarea/12.0.5/PhysicsAnalysis/AnalysisCommon/UserAnalysis/run
athena HelloWorldOptions.py >& hello.log
exit 0
```

- The first line indicates that the job will use sh. pay attention that up until now we used tcsh on physdsk, so now instead of running script.csh we will run script.sh. If you like you can change it to tcsh, but it is not recommended if you wish to run a lot of jobs since tcsh use more memory than sh.

- The line #PBS -N pythiaS indicates that the name of the job is 'pythiaS'. Pay attention that although it seems that this line is commented out, actually it is not and the. The job manager looks for lines like that).
- The line #PBS -q S put the job in a short que. There are three types of queues:
  - X – Very short jobs less than 20 minutes.
  - S – Short jobs less than 8 hours.
  - M – Medium jobs less than 28 hours.
  - L – Long jobs less than 50 hours.
- The line Set –v prints each line executed to the standard output.
- The line cd $PBS_O_WORKDIR changes the directory to the directory you run the job from.

Then you can place the job in the queue:

```
qsub -k eo myjob.sh
```

The switch –k eo tells the job manages to save the stdout and stderr at your home directory. You can see the list of all the jobs by typing:

```
qstat
```

A list of all your jobs:

```
qstat -u username
```

A full description of your jobs:

```
qstat -f -u username
```

You can kill a job using:

```
qdel job-identifier
```

Connecting to the Grid from the PC-Farm

Registration

Consult with Gal on how to get a certificate and install it.

Proxy

You can connect to the Grid only after you get a certificate and regitered to the Atlas Vitrual Organization. In oreder to connect to the grid you first have to login to a computer that runs the User-Interface (UI) middleware. The UI is installed on wipp-ui. So you have to login to wipp-ui computer.

After doing so you can run the proxy as follow (tcsh):

```
setenv GLOBUS_LOCATION /opt/globus
source /opt/globus/etc/globus-user-env.csh
echo password | voms-proxy-init -voms atlas -pwstdin
```

for bash/zsh use:

```
export GLOBUS_LOCATION=/opt/globus
source /opt/globus/etc/globus-user-env.sh
echo password | voms-proxy-init -voms atlas -pwstdin
```

Don Quijote (dq2)

To use dq2_ls, dq2_get, dq2_put, dq2_cr on the PC-Farm: Connect to wipp-ui and run the proxy. To set up the environment:

```
cd /panfs/ppw1/agrp/ohads/dq2/endusers
source setup.csh
cd -
```

You are ready to run dq2.

Running on the Grid

Introduction

The Grid is a collection of CE (Computer Elements) and SE (Storage Elements) in sites around the world coming together to enable distribute analysis and data management in the LHC. There are three types of sites (**Hope I get it right**):

- *Tier 0* – Sites sitting at CERN that get the raw data and transform it to ESD. There will be no direct access to the data at Tier 0. The Tier 0 sites will publish their data to the Tier 1 sites.
- *Tier 1* – Several sites around the world. Will get their data from the Tier 0 sites and build AOD and TAGs. The access to the Tier 1 sites will be only for a physics group collective and not a user access since there will be a need to control the load.
- *Tier 2* – Most of the sites (including our site). Will have the AODs and TAGs.

The Grid has four levels:

- *Application* – Here you build your job or executable (for example an Athena job)
- *Application middleware* – Describes your job requirements using jdl metadata scripts. There are two types of middleware: EDG, gLite.
- *Grid middleware* – Here sit the Resource Broker. It finds the site (or sites) to run your jobs according to the jdl parameters, and according to the location of the dataset you are working on. The main idea is that the Resource Broker will send the job to the data, and by that limiting the amount of data passing through the Grid.
- *Facilities and Fabrics* – Here are the computer clusters (CE and SE) that run the jobs and store the data.

To run jobs on the Grid you must have:

- Certificate – To identify yourself in the Grid.
- VO – You must be belong to a Virtual Organization. In our case we must be registered in the ATLAS VO.
- UI Access – You must have access to a computer running a Grid-UI. (Our own PC-Farm or lxplus)

There are tools to help you find your way on the grid. There is the DDM (Distributed Data Management) tools like DQ2, and GANGA that helps you to manage your processes on the Grid (And in the near future it will have the DQ2 inside it)

Setting up the environment

Certificate, ATLAS-VO

In order to get access to the grid you must get a Certificate and register yourself to the ATLAS VO. In order to do that seek the help of Gal (**Hopefully in the future I find the time to write here the procedure**)

Ask for certificate at the site: https://certificates.iucc.ac.il/cgi-bin/pub/pki?cmd=getStaticPage&name=index
Use Gal's and Lorne's help in that, or follow:
https://twiki.cern.ch/twiki/bin/view/Atlas/WorkBookStartingGrid

Make sure you install the certificate both on you local directory on the pc farm, and on your account on lxplus.

Setting environment variables for the UI

This part is kind of tricky, since they are different versions of UI that works with different Python versions. I will follow here the version that works with GANGA, TNT and DQ2. This is known to works on some of the lxplus nodes (those who weren't upgraded lately) – **I still don't know if it works on our PC-Farm.** I follow here: http://ppewww.physics.gla.ac.uk/~caitrian/tutorial/index.html#pre-tutorial

lxplus

Log in to the specific the specific lxplus node running SLC3 (Scientific Linux Cern version 3) since SLC4 had python collision between Athena and the Grid (will be solved in the near future): lxslc3.cern.ch Then run the following script that will define the necessary environment variables:

```
source /afs/cern.ch/project/gd/LCG-share/previous/etc/profile.d/grid_env.sh
export LFC_HOST='lfc-atlas-test.cern.ch'
export LCG_CATALOG_TYPE=lfc
```

Then You have to start the UI-proxy by executing:

```
voms-proxy-init --voms=atlas
```

The UI-proxy is a copy of your Certificated submitted to the Grid, and it also tells on which VO you are a member of, in this case you tells it that you are a member of ATLAS-VO. The UI-proxy is valid to a limited amount of time (12 hours) and after that it must be renewed; otherwise all your running jobs on the Grid will be terminated. When you start the UI-proxy you can specify the amount of time necessary for your job, and you can also feed the passphrase to the voms-proxy-init from the stdin by running:

```
echo passphrase | voms-proxy-init --voms=atlas --valid h:m --pwstdin
```

Running jobs on the Grid

Running Hello World

Following: https://twiki.cern.ch/twiki/bin/view/Atlas/WorkBookAthenaGrid If you are on lxplus you have to specify it as your UI:

```
source /afs/cern.ch/user/l/lcgatlas/scripts/ATLAS-setenv.sh
```

Copy Hello World job option file to your current directory.

```
cp /afs/cern.ch/atlas/software/releases/AtlasCore/2.0.3/Control/AthenaExamples/AthExHelloWorld/sh
```

Create the script *hello.sh* to be run later:

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

source $VO_ATLAS_SW_DIR/software/12.0.5/setup.sh
source $SITEROOT/AtlasOffline/12.0.5/AtlasOfflineRunTime/cmt/setup.sh


athena.py HelloWorldOptions.py
```

Create a second file hello.jdl which is a **job Description Language**, which are the parameters that defines the job:

```
############ 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-production-12.0.5", other.GlueHostApplicationSoftwareRunTimeEnvir
####################################
```

The 'InputSandbox' is a list of files to be sent with the job. The 'OutputSandbox' is the list of files to be retrieved when the job is finished. The 'Requirements' describes the things your job needs. According to it a proper site will be determined.

Create a UI-proxy and submit your job:

```
grid-proxy-init
edg-job-submit --vo atlas -o jobIDfile hello.jdl
```

You can check the status of your job by typing (you have to follow the menu you get):

```
edg-job-status -i jobIDfile
```

When the job is finished you can retrieve the output files by (you have to follow the menu you get):

```
edg-job-get-output -dir . -i jobIDfile
```

You can cancel a job by typing:

```
edg-job-cancel -i jobIDfile
```

And the following command will give you all the sites with answer the requirements indicated in the jdl file.

```
edg-job-list-match
```

Useful Commands

```
edg-job-submit --vo atlas -o jobIDfile description.jdl
edg-job-status -i jobIDfile
edg-job-get-output -dir . -i jobIDfile
edg-job-cancel -i jobIDfile
edg-job-list-match description.jdl
lcg-infosites --vo atlas ce
lcg-infosites --vo atlas ce
lcg-cr
```

The Commands are:

- Submit the job. Using Virtual Organization (vo) atlas. Output goes to *jobIDfile*. The submitted job uses the *description.jdl* file.
- Check the status of the submitted job, characterized by the *jobIDfile*.
- Download the output files to the local location.
- Cancel a job descriped by *jobIDfile*.
- Prints a list of all the location that are fitted to run the job according to the *description.jdl* file.
- List the sites with CPU availability (ce – CPU Element).
- List the sites with memory availability (se – Storage Element).
- Copy file to a SE (Storage Element). Use --help to see the parameters needed for operating this command.

To get files from cern use (you can use this line exactly to test yourself):

```
lcg-cp -v --vo atlas gsiftp://castorgrid.cern.ch/castor/cern.ch/user/o/osilbert/test.txt file:///
```

DDM – Data Management System

One way to search for dataset is through the AMI (ATLAS Metadata Interface) webpage:
http://lpsc1168x.in2p3.fr:8080/opencms/opencms/AMI/www/index.html

To setup the dq2 environment run:

```
source /afs/usatlas.bnl.gov/Grid/Don-Quijote/dq2_user_client/setup.zsh.CERN
```

See also

- http://marianne.in2p3.fr/datagrid/documentation/EDG-Users-Guide.pdf

This one is especially helpful.

- http://grid-deployment.web.cern.ch/grid-deployment/documentation/LFC_DPM/lcg_util/html/
- http://www.bo.infn.it/alice/introgrd/edg2/node65.html
- http://grid-it.cnaf.infn.it/fileadmin/users/job-submission/job_submission.html
- http://hep-proj-grid-tutorials.web.cern.ch/hep-proj-grid-tutorials/

Also see the Tutorial:
http://jcatmore.web.cern.ch/jcatmore/atlasDCtutorial.html
https://twiki.cern.ch/twiki/bin/view/Atlas/DAonPanda

Running Athena on the Grid

There are few ways for running Athena on the Grid: GANGA, PANDA.

pathena

Look at: https://twiki.cern.ch/twiki/bin/view/Atlas/DAonPanda

In order to use pathena In 12.0.X do the following:

```
cd /somewhere/workarea
cmt co PhysicsAnalysis/DistributedAnalysis/PandaTools
cd PhysicsAnalysis/DistributedAnalysis/PandaTools/cmt
cmt config
source setup.sh
make
cd /somewhere/workarea/.../somedirectory
pathena jobOptions [--site=SITE] [--inDS inputDataSet] --outDS outputDataSet
```

I had a problem with the cvs version that defines the transformation. To silve it I used:

```
cd PhysicsAnalysis/DistributedAnalysis/PandaTools
cvs update -A
cd cmt
cmt config
source setup.sh
make
```

To monitor the result go to:
http://gridui02.usatlas.bnl.gov:25880/server/pandamon/query

Projects

This part is according to my understanding, and I am not sure how accurate it is. Any remarks are welcome.

The Athena framework is divided into projects. Each project is dedicated to a certain goal. When running part of Athena you can use the relevant project and its release compilation.

The project of Athena are (Taken from the Atlas-Workbook):

- *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 LCGCMT project which provides links to LCG supported external software packages such as POOL and ROOT.
- *AtlasEvent*. This project contains the event data model (EDM) and support classes. It dependes 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.
- *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.

In this project I use to reconstruct events using the JobTransforms scripts. Those are scripts that encapsulate inside them the common jobOptions for Atlas production. For more details about them look at: https://twiki.cern.ch/twiki/bin/view/Atlas/AtlasPythonTrf

I show here how I use it (Since I don't really understand it, there are almost no explanations).

First time

Create a working area:

```
cd scratch0
mkdir cmthome
mkdir testarea
mkdir testarea/AtlasProduction-12.0.5
mkdir testarea/AtlasProduction-12.0.5/run
cd cmthome
```

Pay attention that AtlasProduction is not an arbitrary name. The prefix **must** be AtlasProduction, and the suffix must be the version of the release you use. Put in the cmthome the requirements file:

```
set   CMTSITE  CERN
set   SITEROOT /afs/cern.ch
macro ATLAS_DIST_AREA ${SITEROOT}/atlas/software/dist
macro ATLAS_TEST_AREA ${HOME}/scratch0/testarea
use AtlasLogin AtlasLogin-* $(ATLAS_DIST_AREA)
```

Now create the environment

```
source /afs/cern.ch/sw/contrib/CMT/v1r19/mgr/setup.sh
cmt config
```

Every time

```
cd scratch0/cmthome
source ./setup.sh -tag=12.0.5,opt,gcc323
source /afs/cern.ch/atlas/software/builds/AtlasProduction/12.0.5/AtlasProductionRunTime/cmt/setup
export CMTPATH=/afs/cern.ch/atlas/software/releases/AtlasProduction/12.0.5:$CMTPATH
source /afs/cern.ch/atlas/software/releases/AtlasProduction/12.0.5/AtlasProductionRunTime/cmt/set
cd ../testarea/AtlasProduction-12.0.5/run
```

Now you can use

```
trf_ls
```

to see all the JobTransforms scripts.

Generators

Pythia

Here I show how to do it on lxplus, but there is no big difference in the way it is done on physdsk.

Following: https://twiki.cern.ch/twiki/bin/view/Atlas/WorkBookGeneration

If you haven't run Hello World before then you must(???) execute:

```
cd ~/scratch0/testarea/12.0.5
cmt co -r UserAnalysis-00-08-43 PhysicsAnalysis/AnalysisCommon/UserAnalysis
cd PhysicsAnalysis/AnalysisCommon/UserAnalysis/cmt/
source setup.sh
gmake
```

From this section is for everyone. Go to the run directory, get the PDG table file and the job options file.

```
cd ~/scratch0/testarea/12.0.5/PhysicsAnalysis/AnalysisCommon/UserAnalysis/run
get_files PDGTABLE.MeV
get_files jobOptions.pythia.py
```

Edit the job options file in any text editor to be:

```
#--------------------------------------------------------------
# General Application Configuration options
#--------------------------------------------------------------
theApp.setup( MONTECARLO )

include( "PartPropSvc/PartPropSvc.py" )

#--------------------------------------------------------------
# Private Application Configuration options
#--------------------------------------------------------------
theApp.Dlls  += [ "TruthExamples", "Pythia_i" ]
theApp.TopAlg = ["Pythia","DumpMC"]

# --------------------------------------------------------------
# Set output level threshold (2=DEBUG, 3=INFO, 4=WARNING, 5=ERROR, 6=FATAL )
# --------------------------------------------------------------
MessageSvc = Service( "MessageSvc" )
MessageSvc.OutputLevel               = 3

#--------------------------------------------------------------
# Event related parameters
#--------------------------------------------------------------
# Number of events to be processed (default is 10)
theApp.EvtMax = 10

#--------------------------------------------------------------
# Algorithms Private Options
#--------------------------------------------------------------
theApp.ExtSvc += ["AtRndmGenSvc"]
AtRndmGenSvc = Service( "AtRndmGenSvc" )
AtRndmGenSvc.Seeds = ["PYTHIA 4789899 989240512", "PYTHIA_INIT 820021 2347532"]

# Generate Z->ee
Pythia = Algorithm( "Pythia" )
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"]
#
#--------------------------------------------------------------
# Pool Persistency
#--------------------------------------------------------------
#
include( "AthenaPoolCnvSvc/WriteAthenaPool_jobOptions.py" )
theApp.Dlls   += [ "GeneratorObjectsAthenaPoolPoolCnv" ]
Stream1 = Algorithm( "Stream1" )
Stream1.ItemList += [ 'EventInfo#*', 'McEventCollection#*' ]                    #2

Stream1.OutputFile = "pythia.pool.root"
#
##############################################################
```

Now you can change it according to your needs. Some of the important variables inside are:

- *Pythia.PythiaCommand* – In here you put all the Pythia parameters according to the Pythia manual.
- *MessageSvc.OutputLevel* – Determines the log level. You better change it to 3 (INFO) rather than 2 (DEBUG). This will cut a lot of unnecessary log prints.
- *theApp.EvtMax* – Determines the number of events to generate.
- *theApp.TopAlg* – Determines which algorithms to run. The DumpMC is responsible to dump the output of the MC program to the standard output. If you generate a lot of events take it off the list otherwise your log file will be huge.
- *Stream1.OutputFile* – The name of the output file.

After changing the job options file you can run it:

```
athena jobOptions.pythia.py |& tee pythia.log | color
```

Sherpa

Here I show how to do it on phydsk. Unlike Pythia, Sherpa is not part of the Athena framework. As a result the event generation is in two stages: generating the four vectors, and importing it into Athena. Also, since it is not a part of Athena, there is a need to install the package.

I follow here the links: https://twiki.cern.ch/twiki/bin/view/Main/SherpainAthena12
https://twiki.cern.ch/twiki/bin/view/Atlas/HiggsWGCSCExternalGenerators
http://projects.hepforge.org/sherpa/dokuwiki/index

Also, pay attention that version 1.0.8 has a bug in generating b quarks, so make sure you work with version 1.0.9.

Installing Sherpa

Sherpa is installed on the pc-farm (physdsk) in:
/panfs/ppw1/agrp/ohads/SHERPA-MC-1.0.9/SHERPA-1.0.9/Run If you like you can also install it in your own directory, but there is no need to do it.

Download the tarball of Sherpa from their site:
http://projects.hepforge.org/sherpa/dokuwiki/downloads/index

Open the tarball at you desired location. Now you have to compile the package:

```
cd sherpa_dir
./TOOLS/makeinstall -c
```

Now you are ready to use Sherpa

Simulating with Sherpa

To be done….

For now you can use the example for LHC. Pay attention that in the first time you run the process you will get an error since you must compile the libs, after compiling the libs you have to run Sherpa again. The second run has two steps. First Sherpa do the Integration (This can take a while) and then it generates the events:

```
cd sherpa_dir/SHERPA-1.0.9/Run
./Sherpa PATH=LHC/
cd LHC
./makelibs
cd ..
./Sherpa PATH=LHC/
```

Examples for Sherpa processes can be found in:
https://twiki.cern.ch/twiki/bin/view/Atlas/HiggsWGCSCExternalGenerators

I will follow here the example of job 5358 prepared by Markus Warsinski. The configuration files can be found at the following tar ball. Extract the tar ball into a new directory. The important files are:

- Run.dat - This files include parameters that defines parameters for the specific run, like:
  - EVENTS - Defines the number of events to generate
  - OUTPUT - Defines the output level
  - SHERPA_OUTPUT - The prefix of the file name to give the output files
  - FILE_SIZE - Number of events at each file
- Selector.dat - Defines cuts on the generated process
- Fragmentation.dat - Defines the way to do the Fragmentation. In this example, since Tau decays handled like Hadron decays, A new Decay library was defined and the Tau decay was changed to be only leptonic decay. The parameter DECAYPATH points to the decay library. and in this library there is a directory TauDecay that defines all the decay modes of the Tau. The file TauDecay.dat was changed to have only the leptonic decay.
- Hadron.dat - Defines the different Hadrons
- Particle.dat - Defines the fields in the theory. In this example the mass and the width of the SM higgs was changed to be similiar to that of the A/H in the MSSM
- Model.dat - Defines the parameters of the model. In this example the yukawa couplings to the light quarks were turned off since this configuration was dedicated to bbA->tau tau process
- Processes.dat - Defines the desired processes to generate. You can add the parameter Print_Graphs to each of the processes to tell Sherpa to create Feynamnn graphs of the generated process in TeX format.

There are two important directories:

- Process - In this directory Sherpa stores the compiled libs. When changing the process or the model you must delete the content of this directory and run Sherpa, make the libs and then run Sherpa again. In this library you can find the Feynman graphs (If you oredered Sherpa to print them)
- Results - You can store the results of the integration part in this directory using the parameter RESULT_DIRECTORY=/path_to_dir/. This will save time at future runs of Sherpa. After changing processes you must delete the content of this directory and let Sherpa do the integration again.

To run the example you have to be in the Run directory of sherpa and execute:

```
./Sherpa PATH=exampledir/ OUTPUT_PRECISION=11 RESULT_DIRECTORY=exampledir/Results/
```

Importing Sherpa files into Athena

To be done later… For now, following:
http://atlas-computing.web.cern.ch/atlas-computing/links/buildDirectory/AtlasSimulation/2.0.4/Generators/ReadSherpa

Get an example jobOptins file and example evnt file (or use the evnt file you generated with Sherpa) and run Athena:

```
get_joboptions jobOptions.sherpa.py
get_files sherpa_Z_nunu.evts
athena.py jobOptions.sherpa.py |& color
```

You can use *jobOptions.sherpaToCbnt.py* to produce CBNT file.

Generators & Calculators: (Events, Cross Sections, Widths, Couplings etc.)

All these calculators can be run over linux and unix (details on their website). In addition I succeeded to run them over WinXp after installing cygwin.

CPSuperH: http://www.hep.man.ac.uk/u/jslee/CPsuperH.html
FeynHiggs: http://www.feynhiggs.de/
MCFM: http://mcfm.fnal.gov/

Using ROOT

ROOT is analysis framework, it consist a C++ interpreter and a collection of C++ packages. It is very eassy to use ROOT in the linux environment using gnu C++ compiler. To use ROOT on WinXp using MSVC 7.1 or above you have to install the complete cygwin environment. cygwin can be downloaded from:
http://www.cygwin.com/
To configure the Microsoft Visual Studio environment to work with root follow the very helpfull links:
http://www.muenster.de/~naumana/rootmsvc.html
http://gentit.home.cern.ch/gentit/rootandvisual7/cas/XPDebug.html

When trying to open graphic windows from ROOT you must define a TApplication object in the program, and you must run the cygwin X server.

Documentation of the different classes in ROOT can be found at:
http://root.cern.ch/root/html/ClassHierarchy.html
and a user guide can be found at:
http://root.cern.ch/root/html/ClassHierarchy.html

Using Tree Friend

It is convinient to use tree friends when dumping data in HighPtView into different files (for example trigger/Truth/Reco). In the following example The CollectionTree has inside keys for the Truth tree and the Reco tree.

```
TChain * ch_Reco = new TChain("RecoTree");
ch_Reco->Add("RecoFiles*");
TChain * ch_CT = new TChain("CollectionTree");
ch_CT->Add("CTFiles*");
TChain * ch_Truth = new TChain("TruthTree");
ch_Truth->Add("TruthFiles*");
ch_CT->AddFriend(ch_Truth);
ch_CT->AddFriend(ch_Reco);
ch_CT->Draw("var1");
ch_CT->Draw("RecoTree.var2-TruthTree.var3");
```

Bibliography

https://twiki.cern.ch/twiki/bin/view/Atlas/WorkBook
http://www.hep.ucl.ac.uk/atlas/atlfast/
https://twiki.cern.ch/twiki/bin/view/Atlas/SusyView
http://cvs.web.cern.ch/cvs/howto.php#accessing
https://twiki.cern.ch/twiki/bin/view/Atlas/AtlasPythonTrf
https://certificates.iucc.ac.il/cgi-bin/pub/pki?cmd=getStaticPage&name=index
http://grid-deployment.web.cern.ch/grid-deployment/documentation/UI-lxplus/
https://twiki.cern.ch/twiki/bin/view/Atlas/TriesteAnalysisTutorial1203
https://twiki.cern.ch/twiki/bin/view/Main/SherpainAthena12
https://twiki.cern.ch/twiki/bin/view/Atlas/HiggsWGCSCExternalGenerators
http://projects.hepforge.org/sherpa/dokuwiki/index
http://atlas-computing.web.cern.ch/atlas-computing/links/buildDirectory/AtlasSimulation/2.0.4/Generators/ReadSherpa

---

-- Remarks, suggestions and corrections Ohad Silbert

This topic: Main > WISComputingGuide
Topic revision: r25 - 2008-10-28 - OhadSilbert

Copyright &© 2008-2020 by the contributing authors. All material on this collaboration platform is the property of the contributing authors.
Ideas, requests, problems regarding TWiki? Send feedback