

## xAOD EDM

[Introduction](#)[Setting up](#)[xAOD EDM software](#)[Browsing the xAOD with the TBrowser](#)[Interactive ROOT with the xAOD](#)[PyROOT with the xAOD](#)

## Introduction

This short hands-on tutorial should be completed after the xAOD EDM lectures and hearing about the information available in the xAOD. You now have the opportunity to use ROOT to quickly look at information in the xAOD. Complete analysis examples are shown in other parts of the tutorial series.

The tutorial below has been migrated to use Analysis Release 21.2 (which is a release using **git+cmake**).

## Setting up

We will work on lxplus for these examples.

From an linux machine you do it with:

```
ssh -X lxplus.cern.ch
```

From a MacOS X laptop you do:

```
ssh -Y lxplus.cern.ch
```

Make sure you have X11 forwarding or you won't be able to open the TBrowser from ROOT. (Ask us if you need help with this.)

### Using the Analysis Release

We will use the ATLAS [Analysis Release](#) to setup our ROOT environment and other packages (xAOD EDM and tools). To simply browse the xAOD in ROOT you only need a new-ish version of ROOT (the default with the Analysis Base should be fine, but the newer the better), but to work interactively or using macros (described below) you need these additional 'other' packages automatically included in the Analysis Release setup. This Analysis Release will be described in more detail later in this tutorial (when we learn about doing analysis in ROOT). We will show you here how to setup the Analysis Release on lxplus, but briefly:

- it sits on cvmfs (so can be used on other sites with cvmfs access, like the Grid sites)
- has support in SLC6 and MacOS (need to specify more details)
- has a size of O(300 MB)
- can be checked out from cvmfs in full and run on one of the above supported platforms

It is recommended you create a directory from where you will setup the Analysis Release. So let's create a working directory, call it **TutorialEDM**, and navigate there:

```
mkdir TutorialEDM
```

```
cd TutorialEDM
```

Now we will source a script to be able to access the ATLAS software:

```
setupATLAS
```

This alias is already defined for you when you log in to lxplus. After you type it you will see a list of commands you can type to setup various ATLAS computing tools.

If you are **not** working on lxplus: you may need to define these variables (try typing `setupATLAS` and see if that alias is already defined on your system):

```
export ATLAS_LOCAL_ROOT_BASE=/cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase
alias setupATLAS='source ${ATLAS_LOCAL_ROOT_BASE}/user/atlasLocalSetup.sh'
setupATLAS
```

Now let's setup our Analysis Release, we will use a 'flavor' called AnalysisBase (which is the general purpose one maintained by ASG). The instruction below refers to using analysis release 21.2 (which is intended for xAODs produced with the Athena Release 21, compiled using cmake, and managed using git):

```
asetup AnalysisBase,21.2.93
```

The instructions above are not complete for doing a proper analysis, but for this quick demonstration of browsing the xAOD they will be enough. Complete instructions for setting up a releases can be found here: [Physics Analysis Workbook R21: Setting up the Analysis Release](#)

 Be sure to check the AnalysisBase release used above is the latest version used in the tutorial earlier --> `asetup AnalysisBase,21.2.93`

We will define the environment variable `$ALRB_TutorialData` which will point to the location of the input data used for the tutorial (this depends on which tutorial and where you are running). This variable may be provided to you by the tutorial organizers, but if you are working from lxplus you can do one of the following (depending on which shell you are using):

Tutorial	zsh, bash shells	csh shell
<a href="#">CERN January 2018</a> (recommended)	export ALRB_TutorialData=/afs/cern.ch/atlas/project/PAT/tutorial/cern-jan2018/	setenv ALRB_TutorialData /afs/cern.ch/atlas/project/PAT/tutorial/cern-jan2018/
<a href="#">CERN September 2017</a>	export ALRB_TutorialData=/afs/cern.ch/atlas/project/PAT/tutorial/cern-sept2017/	setenv ALRB_TutorialData /afs/cern.ch/atlas/project/PAT/tutorial/cern-sept2017/
<a href="#">CERN July 2017</a>	export ALRB_TutorialData=/afs/cern.ch/atlas/project/PAT/tutorial/cern-july2017/	setenv ALRB_TutorialData /afs/cern.ch/atlas/project/PAT/tutorial/cern-july2017/
<a href="#">CERN October 2016</a> (recommended if using 2.4.X or earlier)	export ALRB_TutorialData=/afs/cern.ch/atlas/project/PAT/tutorial/cern-oct2016/	setenv ALRB_TutorialData /afs/cern.ch/atlas/project/PAT/tutorial/cern-oct2016/
<a href="#">CERN June 2016</a>	export ALRB_TutorialData=/afs/cern.ch/atlas/project/PAT/tutorial/cern-june2016/	setenv ALRB_TutorialData /afs/cern.ch/atlas/project/PAT/tutorial/cern-june2016/
<a href="#">UChicago/ANL March 2016</a>	export ALRB_TutorialData=/afs/cern.ch/atlas/project/PAT/tutorial/chicago-mar2016/	setenv ALRB_TutorialData /afs/cern.ch/atlas/project/PAT/tutorial/chicago-mar2016/
<a href="#">CERN February 2016</a>	export ALRB_TutorialData=/afs/cern.ch/atlas/project/PAT/tutorial/cern-feb2016/	setenv ALRB_TutorialData /afs/cern.ch/atlas/project/PAT/tutorial/cern-feb2016/
<a href="#">CERN May 2015</a>	export ALRB_TutorialData=/afs/cern.ch/atlas/project/PAT/tutorial/cern-may2015/	setenv ALRB_TutorialData /afs/cern.ch/atlas/project/PAT/tutorial/cern-may2015/

Now you can use this environment variable instead of the full path to the files (this allows the tutorial instructions to be site independent). Note, the environment variable \$ALRB\_TutorialData can also be set in ~/.bashrc for the purpose of this tutorial in the same way as \$RUCIO\_ACCOUNT above.

xAOD EDM software

For the moment the best way to understand what variables you can access for each object, you should look at the code.

The EDM code can be browsed online here: [athena/Event/xAOD \(git\)](#). In most cases, it should be clear which package in the structure to look at. For example if you are interested in taus, you would look at the [xAODTau](#) package. For each xAOD EDM package you can either choose a particular tag of that package (corresponding to the version in your Athena or Analysis Release versions), or you can look in the master version to see the latest version.

Browsing the xAOD with the TBrowser

Now let's open the xAOD in ROOT, and open a TBrowser for browsing:

```
root $ALRB_TutorialData/r9315/mc16_13TeV.410501.PowhegPythia8EvtGen_A14_ttbar_hdamp258p75_nonallhad.merge.AOD.e5458_s3126_r9364_r9315/AOD.11182705_000001.pool.root.1
root[0] TBrowser b;
```

You will get lots and lots of warnings about dictionaries not available, don't worry about those for now. The TTree containing the variables is called `CollectionTree`. Feel free to open that TTree and play with the items inside.

Can you find and understand the variables we discussed in the lectures?

Can you see how the variables are organized?

Look and compare containers with and without `Aux`, representing the Auxillary store. When looking at the xAOD in the TBrowser we need to be aware of this, but when working interactively, or with a macro, or a full-blown compile analysis, we will interact with the xAOD objects through an interface and don't need to explicitly worry about this Auxillary store business.

Interactive ROOT with the xAOD

Let's use interactive ROOT (xAODRootAccess) to look at the xAOD. Open root in your terminal and let's tell ROOT we are using an xAOD input, create a (transient) TTree, and draw a simple histogram (it might take a minute or two to draw the histogram):

```
root [0] xAOD::Init();
root [1] f = TFile::Open( "$ALRB_TutorialData/r9315/mc16_13TeV.410501.PowhegPythia8EvtGen_A14_ttbar_hdamp258p75_nonallhad.merge.AOD.e5458_s3126_r9364_r9315/AOD.11182705_000001
root [2] t = xAOD::MakeTransientTree(f)
root [3] t->Draw( "Electrons.pt0 - Electrons.trackParticle0.pt0" );
```

It is super important that you do `xAOD::Init()` **before** loading the file. You will learn more about these methods (from xAODRootAccess) in the EventLoop analysis session covered later in the tutorial.

Also, once you've made the transient tree you can double click things nicely in the TBrowser by going to the following folder:

```
root/Root Memory/CollectionTree
```

Although I noticed in ROOT6 (6.02.05) when you actually try to double click on branches this produces a segmentation violation, but will still plot your quantity anyway.

PyROOT with the xAOD

PyROOT is very handy when you want to run a quick macro (it's also fun for bigger analyses too). Here we will create a little PyROOT macro to print to screen some xAOD quantities. With your favorite editor create and open the new python script, xAODPythonMacro.py, and fill it with this content:

```
#!/usr/bin/env python

# Import ROOT:
import ROOT

# Initialize the xAOD infrastructure:
if(not ROOT.xAOD.Init().isSuccess()): print "Failed xAOD.Init()"

# Set up the input files:
tutorialPath = "$ALRB_TutorialData"
fullTutorialPath = ROOT.gSystem.ExpandPathName(tutorialPath)
fileName = fullTutorialPath + "/r9315/mc16_13TeV.410501.PowhegPythia8EvtGen_A14_ttbar_hdamp258p75_nonallhad.merge.AOD.e5458_s3126_r9364_r9315/AOD.11182705_000001.pool.root"

treeName = "CollectionTree" # default when making transient tree anyway

f = ROOT.TFile.Open(fileName)

# Make the "transient tree":
t = ROOT.xAOD.MakeTransientTree( f, treeName)

# Print some information:
print( "Number of input events: %s" % t.GetEntries() )
for entry in xrange( 0,100): # let's only run over the first 100 events for this example
    t.GetEntry( entry )
    print( "Processing run #%, event #%" % ( t.EventInfo.runNumber(), t.EventInfo.eventNumber() ) )
    print( "Number of electrons: %i" % len( t.Electrons ) )
    # loop over electron collection
    for i in xrange( t.Electrons.size()):
        el = t.Electrons.at(i)
        print( " Electron trackParticle eta = %g, phi = %g" % ( el.trackParticle().eta(), el.trackParticle().phi() ) )
        pass # end for loop over electron collection
    pass # end loop over entries
# clear transient trees to avoid crash at end of job
ROOT.xAOD.ClearTransientTrees()
```


Let's make the script executable: `chmod +x xAODPythonMacro.py`, and now run it:

```
./xAODPythonMacro.py
```

Hopefully it runs and you get some xAOD information printed to the screen.

Major updates:  
-- [LouiseHeelan](#) - 27 Feb 2014

Responsible: [LouiseHeelan](#)  
Last reviewed by: **Never reviewed**

I	Attachment	History	Action	Size	Date	Who	Comment
	<a href="#">Screen_Shot_2014-10-14_at_13.54.53.png</a>	r1	<a href="#">manage</a>	279.3 K	2014-10-14 - 13:56	<a href="#">AttilaKrasznahorkay</a>	<a href="#">RootCore</a> project loaded into Xcode

Topic revision: r83 - 2019-10-18 - [AdamJacksonParker](#)