Table of Contents

HarvardAnalysisFramework

This page is obsolete. Please go to MuonIDAnalysis.

The Harvard Analysis Framework is a combination of Athena and ROOT code designed for quick and efficient analysis of ESD's, with specific focus on muon related quantities. It is designed to be lightweight, easy to run, update, and share code, and have room to grow as more information is needed and the amount of data collected by ATLAS increases.

Quick Start

This section contains very brief instructions on how to get up and running as quickly as possible. For more information, please see later sections.

Making Ntuples

The Harvard Analysis Framework is designed to be run in release 15.6.3, so start by setting up your environment:

```
mkdir -p ~/testarea/15.6.3/run
source ~/cmthome/setup.sh -tag=15.6.3
cd ~/testarea/15.6.3
```

(Note that your setup may look slightly different - see the Computing workbook for more info)

Next, check out and compile the ntuple making package:

```
cmt co ${SVNGRP}/Institutes/Harvard/AthenaAnalysis/NtupleMakers
cd NtupleMakers/cmt
gmake
```

Finally, grab the example job options file, and run it!

```
cd ../../run
get_files -jo NtupleMaker_Example.py
athena NtupleMaker_Example.py
```

By default, the job runs on a DESD from the 2009 collisions that's on CERN AFS. If you don't have access to CERN AFS, please change the InputCollections at the beginning of the file.

Once the job is finished, you should an ntuple called `ntuple.root` in your run directory!

Adding Info to the Ntuple

Lets say you've made your first ntuple, but there's something missing that you need for an analysis. As an example, lets say that you're unhappy that TrackDumper stores the track theta, and you'd like to add the track eta.

(A note - all the line numbers here refer to NtupleMakers-00-00-15 which is available here. If you've checked out a more recent version, the line numbers might be slightly different. If you'd like to follow along, please use the links below, or check out that version of NtupleMakers with the command

```
cmt co -r NtupleMakers-00-00-15 ${SVNGRP}/Institutes/Harvard/AthenaAnalysis/NtupleMakers
cd NtupleMakers/cmt
gmake
```

Note that it probably very easy to figure out the exact line on the head version if you'd prefer to do it that way)

Start by looking at the header file for TrackDumper here. Specifically, look at line 64 - that's where we declare the vector to hold track theta:

```
std::vector *m_theta;
```

we'll need to also have a vector for track eta, so right after that line, add a similar line:

```
std::vector *m_eta;
```

Now that we've declared it, we have to add it to the ntuple and fill it!

Look at the src file for TrackDumper here. Specifically, look at line 113:

```
addBranch("track_theta", m_theta);
```

That's how we add the variable to the ntuple - the first argument is the name in the ntuple and the second is the vector that holds the info. So add a new line after that one that's similar:

```
addBranch("track_eta", m_eta);
```

Now look at line 157:

```
m_theta->clear();
```

This code is called every event before execution and insures when you write out the tracks for event 1053, you don't still have info left over from tracks from event 1052.

Lastly, look at line 260:

```
m_theta->push_back(origin_theta);
```

This is where you fill the information. In this case, we calculate the theta at the origin. For the purposes of this tutorial, we'll just cheat and calculate the eta from the theta, so add this line right after:

```
m_eta->push_back(-1. * ln ( tan(origin_theta/2.) ));
```

And now you're all set! Compile the code and run it as above, and your ntuple will now contain a variable called track_eta!

Examining the ntuples in ROOT

The easiest way to examine the ntuples in ROOT is whatever way you're most comfortable with. You can browse the ntuple as usual with:

```
root -l ntuple.root
TBrowser b;
```

You can call make class or make selector on it like with a normal ntuple.

Lastly, if you'd like to use the ROOT framework designed to work with the ntuple, you can run an example job in the following way:

First, checkout and build the ROOT framework:

```
cd ~/testarea/15.6.3
cmt co ${SVNGRP}/Institutes/Harvard/RootAnalysis
cd RootAnalysis
./build
```

Run the analysis example:

```
cd Analysis/AnalysisExample
root -l run.C
```

You can change the input ntuple by altering `run.C` . It currently points to an ntuple available on CERN AFS, so if you don't have access to CERN AFS, you should change the input before running.

To see a more sophisticated example, take a look at, for example, JPsi, which looks for two track resonances and uses some of the tricks available from the framework.

Athena Framework

The athena framework consists of a collection of algorithms called dumpers, each of which dump information about a single (or closely related) athena object. This section describes why this approach was chosen, how the dumpers work, and what information they currently put in the ntuple.

Framework design

The HarvardAnalysisFramework had three basic design goals

- Begin analysis quickly
- Ability to develop code in parallel
- Room to grow the framework as data quantity and complexity increases

To achieve these goals, we made a number of design choices. First, we chose to make the framework be as similar as possible to existing code while still achieving our objectives. This meant it was easy to include existing code and was easy to learn how to make new code for the framework. Second, we chose to make the framework as modular as possible by breaking it into small, distinct pieces. This allows developers to work in parallel without fear of creating conflicts. It also allows users to easily add or remove information based upon their analysis needs. Third, we chose to write the code in such a way as to promote good bookkeeping about versioning and input files. Finally, we chose to reduce the size of the ntuples whenever possible, in order to decreases disk space and speed up analysis.

There are three basic parts to the athena framework

- NtupleMakerBase is a base class from which all ntuple dumpers derive. This class handles basic data flow as well as performs bookkeeping.
- A dumper is an athena algorithm that fills the ntuple with information about a basic athena object (or collection of similar objects).
- A job configuration is a job options file specifying which algorithms should be run and in what order.

These three parts are described below

NtupleMakerBase

The NtupleMakerBase class (src, header) is a base class for all ntuple dumpers in the framework. It was chosen to be very similar to the common ntuple making base class CBNT_AthenaAwareBase (which it inherits from) in order to allow ease of code transition and little time required to learn the new framework.

The basic concepts are very similar: for each variable you wish to fill in your ntuple, you declare memory in your class to hold the information; you then add a branch for that variable in the ntuple, and fill it as desired. However, there are a few differences in the flow of an algorithm in this framework.

Instead of an initialization function, NtupleMakerBase offers three functions in order to promote better coding:

```
virtual StatusCode loadServices()
virtual StatusCode loadTools()
virtual StatusCode addBranches()
```

The first two functions are overloaded in child classes to load services and tools, respectively, while the last is used to add branches to the ntuple. For example:

```
StatusCode TrackDumper::loadTools() {

    StatusCode sc = m_trackSummaryTool.retrieve();
    if(sc.isFailure()) {
        ATH_MSG_WARNING("Unable to load summary tool!");
        m_trackSummaryTool = 0;
    }


    return StatusCode::SUCCESS;

}

StatusCode TrackDumper::loadServices() {

    StatusCode sc;
    sc = this->detStore()->retrieve(m_sctMgr, "SCT");
    if(sc.isFailure() || !m_sctMgr) {
        ATH_MSG_FATAL("Unable to retrieve SCT detector manager");
        return sc;
    }

    return StatusCode::SUCCESS;

}

StatusCode TrackDumper::addBranches() {

    addBranch("track_ntracks", m_NumberOfTracks,"nTracks/i");
    addBranch("track_index", m_index);
    addBranch("track_qoP", m_qOverP);
    addBranch("track_phi", m_phi);
    addBranch("track_theta", m_theta);
    addBranch("track_d0",   m_d0);
    addBranch("track_z0",   m_z0);

    return StatusCode::SUCCESS;

}
```

From the example, you can see how branches are added to the ntuple:

```
    addBranch("track_phi", m_phi);
```

The first argument is the name of the branch in the ntuple and the second is the location in memory of the information (such as a `std::vector*` in this case).

To clear branches between events, NtupleMaker base offers the function to be overloaded:

```
virtual StatusCode clearBranches()
```

For example:

```
StatusCode TrackDumper::clearBranches() {

    m_NumberOfTracks = 0;
    m_index->clear();
    m_qOverP->clear();
    m_phi->clear();
    m_theta->clear();
    m_d0->clear();
    m_z0->clear();
```

```
        return StatusCode::SUCCESS;

    }
```

During execution, NtupleMakerBase offers four functions to be overloaded:

```
virtual StatusCode processNewJob()
virtual StatusCode processNewRun()
virtual StatusCode processNewBlock()
virtual StatusCode processNewEvent()
```

The first is called only upon loading the first event of the job, the second whenever a new run is encountered, the third whenever a new lumiblock is encountered, and the last every event. This offers very simple ways for filling metadata or other data that need not change every event. For example:

```
    StatusCode TriggerDumper::processNewBlock() {

        // add the prescales to the ntuple
        for(std::vector<std::string>::const_iterator iter = m_configuredTriggers.begin();
                iter != m_configuredTriggers.end(); ++iter)
            m_aan_triggerPrescales->push_back(m_trigDecisionTool->getPrescale(*iter));

        return StatusCode::SUCCESS;

    }
```

dumps trigger prescales only every lumiblock (as they cannot change otherwise) rather than every event. Correctly placing data dumping can dramatically improve performance and decrease ntuple size.

Lastly, for finalizing, NtupleMakerBase offers a function for overloading:

```
virtual StatusCode finish()
```

In addition to the above, NtupleMakerBase also provides some bookkeeping by marking the ntuple with algorithms used and their version. Specifically, every ntuple will have a branch for each algorithm run of the form

```
ALGNAME_VERSION
```

For example, if TriggerDumper is run from the version of the code NtupleMakers-00-00-15, it will tag the ntuple with a branch called

```
TriggerDumper_NtupleMakers_00_00_15
```

Dumpers

Each dumper is responsible for dumping the information about a single athena object or a collection of closely related athena objects. This breakdown of the work offers a number of benefits

- It reduces conflicts from developers making changes on the same file at the same time
- It provides a primary contact or person responsible for each piece of code
- It allows users to customize exactly what information they want in their ntuple

A list of dumpers is available here

Currently, they are:

- BookKeeper

- CaloDumper
- EGammaDumper
- ElectronDumper
- LumiDumper
- METDumper
- MuonDumper
- MuonSegmentDumper
- PhotonDumper
- TrackDumper
- TriggerDumper
- V0Dumper
- VxDumper

A more detailed description of the contents of each dumper is available in later sections

Job Configuration

The job is configured by a job options file. This must do two things:

- It must configure the detector services and tools correctly so they are available for the job
- It must add the desired dumpers to the job

The current job options is available here

To add a dumper to the job, import the algorithm from the NtupleMaker configurable, configure the desired properties, and then add it to the job sequence. For example:

```
from NtupleMakers.NtupleMakersConf import MuonSegmentDumper
MuonSegmentDumper = MuonSegmentDumper()
MuonSegmentDumper.EventInfoKey = EventInfoKey
MuonSegmentDumper.NtuplePrefix = "seg"
MuonSegmentDumper.MuonSegmentContainerKey = "MooreSegments"
Sequencer += MuonSegmentDumper
```

The first line loads the MuonSegmentDumper class The second creates an instance The next three lines configure properties of the class The last line add its it to the job.

Ntuple contents

In this section of the contents of the ntuple as dumped by each dumper are described in detail

Book Keeping

Book keeping is performed by the BookKeeper algorithm (src, header). It contains information about the job structure.

Default name	Filled on	Description
job_tag	Job	Job tags (release number, geometry, etc.)
job_inputfiles	Job	Physical file name of input files
block_number	Block	Current luminosity block

Calorimeter

Calorimeter information is dumped by the CaloDumper algorithm (src, header). It contains information about calo clusters and calo towers

Default name	Filled on	Description
caloCluster_nClusters	Event	Number of calo clusters in event
caloCluster_totE	Event	Total energy in calo clusters
caloCluster_eta	Event	Calo cluster eta
caloCluster_phi	Event	Calo cluster phi
caloCluster_hadEnergy	Event	Calo cluster hadronic energy
caloCluster_emEnergy	Event	Calo cluster EM energy
caloCluster_em0	Event	Calo cluster EM energy in layer 0
caloCluster_em1	Event	Calo cluster EM energy in layer 1
caloCluster_em2	Event	Calo cluster EM energy in layer 2
caloCluster_em3	Event	Calo cluster EM energy in layer 3
caloCluster_had1	Event	Calo cluster hadronic energy in layer 1
caloCluster_had2	Event	Calo cluster hadronic energy in layer 2
caloCluster_had3	Event	Calo cluster hadronic energy in layer 3
caloCluster_larB	Event	Calo cluster energy in LAr barrel
caloCluster_larE	Event	Calo cluster energy in LAr endcap
caloCluster_fCal	Event	Calo cluster energy in forward calorimeter
caloCluster_hadE	Event	Calo cluster energy in hadronic endcap
caloCluster_tileB	Event	Calo cluster energy in tile barrel
caloCluster_tileG	Event	Calo cluster energy in tile gap
caloCluster_tileExB	Event	Calo cluster energy in tile extended barrel
caloTower_nTowers	Event	Number of calo towers in event
caloTower_totE	Event	Total energy in calo towers
caloTower_energy	Event	Calo tower energy
caloTower_eta	Event	Calo tower eta
caloTower_phi	Event	Calo tower phi

Egamma

Information about Egamma objects is dumped by three dumpers. First, egammaDumper (header, icc) dumps information that similar for both electron and photon objects. Then, electron specific info is dumped by ElectronDumper (src, header) and photon specific info is dumped by PhotonDumper (src, header)

EgammaDumper

Default name	Filled on	Description
eg_et	Event	egamma transverse energy
eg_eta	Event	egamma eta
eg_phi	Event	egamma phi
eg_deteta	Event	egamma eta (at calorimeter)
eg_ethad1	Event	egamma transverse energy in first layer of hadronic calorimeter
eg_hadleak	Event	egamma transerse energy in hadronic calorimeter
eg_f1	Event	egamma fraction of total energy in the first of the three samplings
eg_isocone2	Event	egamma isolation transverse energy in cone of 0.2
eg_isocone4	Event	egamma isolation transverse energy in cone of 0.4
eg_fracs1	Event	

		egamma ratio of energy outside of core (but in seven strips) to energy inside shower core (three strips)
eg_deltaemax2	Event	egamma mysterious "parametrization of E(2nd max)"
eg_deltae	Event	difference between 2nd maximum and energy minimum between it and the maximum
eg_demaxs1	Event	relative difference in magnitude between first and second energy maxima
eg_wtot	Event	egamma sum runs over all strips in a width of d_eta x d_phi = 0.0625 x 0.2, roughly 40 strips across
eg_weta1	Event	egamma like wtot, but summing over only three strips in making the RMS
eg_e277	Event	egamma energy in a 7x7 (in eta x phi, cell unit width is 0.025 x 0.0245) cluster
eg_reta37	Event	egamma e277/e237
eg_rphi33	Event	egamma e233/e237
eg_weta2	Event	egamma width of shower in second sampling

Electron

Default name	Filled on	Description
ele_isTight	Event	Does electron pass isTight pid requirements
ele_isMedium	Event	Does electron pass isMedium pid requirements
ele_isLoose	Event	Does electron pass isLoose pid requirements
ele_charge	Event	electron charge
ele_trkindex	Event	Index of ID track (for linking with info from TrackDumper)
ele_trkz0	Event	electron ID track z0

Photon

Default name	Filled on	Description
pho_isTight	Event	Does photon pass isTight pid requirements
pho_isLoose	Event	Does photon pass isLoose pid requirements
pho_isConv	Event	Is the photon flagged as a conversion
pho_conv_vtx_ntracks	Event	Number of tracks coming from photon conversion vertex
pho_conv_vtx_R	Event	Radius of conversion vertex postion
pho_conv_vtx_phi	Event	Phi of conversion vertex position
pho_conv_vtx_z	Event	z position of conversion vertex position

Jets

Jets information is dumped by the JetDumper algorithm (src, header). It contains information about each JetCollection specified in the JobOptions.

Default name	Filled on	Description
JetCollectionName_nJets	Event	Number of jets in event
JetCollectionName_e	Event	Energy of Jet in GeV
JetCollectionName_px	Event	Momentum of Jet in the x direction in GeV
JetCollectionName_py	Event	Momentum of Jet in the y direction in GeV
JetCollectionName_pz	Event	Momentum of Jet in the z direction in GeV
JetCollectionName_pt	Event	Transverse momentum of Jet in GeV
JetCollectionName_eta	Event	Eta of Jet
JetCollectionName_phi	Event	Phi of Jet
JetCollectionName_Ncon	Event	Number of constituents in a Jet
JetCollectionName_ptcon	Event	Transverse momentum of constituent in GeV

JetCollectionName_econ	Event	Energy of constituent in GeV
JetCollectionName_etacon	Event	Eta of constituent
JetCollectionName_phicon	Event	Phi of constituent
JetCollectionName_weightcon	Event	Weight of constituent
JetCollectionName_emfrac	Event	ElectroMagnetic fraction of Jet energy
momentList	Job	Jet moments (specified in JobOptions)
JetCollectionName_moments	Event	Moments of Jet
JetCollectionName_mu_Nmu	Event	Number of Muons associated with Jet
JetCollectionName_mu_pt	Event	Pt of Muon associated with Jet in GeV
JetCollectionName_mu_eta	Event	Eta of Muon associated with Jet
JetCollectionName_mu_phi	Event	Phi of Muon associated with Jet
JetCollectionName_mu_charge	Event	Charge of Muon associated with Jet
JetCollectionName_mu_d0	Event	d0 of Muon associated with Jet
JetCollectionName_mu_z0	Event	z0 of Muon associated with Jet

Missing Energy

Information about the event missing energy is dumped by the METDumper (src, header).

Default name	Filled on	Description
mu_name	Event	Name of MET algorithm
met_metx	Event	MET in x direction
met_mety	Event	MET in y direction
met_source	Event	Detectors/Objects used for MET calculation

Muons

Muon information is dumped by the MuonDumper algorithm (src, header) for muon tracks and hits, and by the MuonSegmentDumper algorithm (src, header) for muon segments.

Naming convention

The muon variable names follow the D3PD scheme, to permit easier re-use of muon ntuple analysis code between the HarvardAnalysisFramework and MuonD3PDMaker.

Track information is filled for combined, muon spectrometer and inner detector muons. The corresponding variable affixes are:

Track kind	Affix
Combined	-
Muon Spectrometer	ms
Inner Detector	id

For each type of track (muon spectrometer, cobmined, inner detector), track parameters are filled relative to the following locations:

Extrapolation location	suffix
Default perigee	-
Origin	ex0
Primary vertex	exPV
BI	exBI
BM	exBM

BO	exBO
Muon Spectrometer entrance	exMS

Combined Muon information

Default name	Filled on	Description
mu_hasCombinedMuonTrackParticle	Event	whether the muon has a combined TrackParticle
mu_numberOfSegments	Event	Number of muon segments for muon
mu_matchchi2	Event	chi2 of match between ID and MS
mu_bestMatch	Event	whether the combined muon is the best match to an MS muon
mu_author	Event	Primary muon author
mu_isLoose	Event	Test loose muon
mu_isMedium	Event	Test medium muon
mu_isTight	Event	Test tight muon
mu_etcone40	Event	Calorimeter isolation energy in cone of size 0.4
mu_alsoFoundByCaloMuonId	Event	Whether seen by calo muon id
mu_energyLoss	Event	Energy loss in the calorimeter

Track parameters/errors

Default name	Filled on	Description
mu_pt	Event	Transverse momentum in GeV (combined muon)
mu_qoverp	Event	Charge divided by magnitude of momentum (combined muon)
mu_eta	Event	Eta (combined muon)
mu_phi	Event	Azimuthal angle (combined muon)
mu_d0	Event	Transverse impact parameter (combined muon)
mu_z0	Event	Longitudinal impact parameter (combined muon)
mu_vx	Event	perigee x coordinate (combined muon)
mu_vy	Event	perigee y coordinate (combined muon)
mu_vz	Event	perigee z coordinate (combined muon)

- These are replicated for MS and ID tracks. Hence, mu_id_pt is the transverse momentum for the ID track while mu_ms_eta gives the eta of the spectrometer track
- The above are also filled for extrapolated parameters. Hence, mu_id_vx_exMS is the extrapolated perigee for the ID track extrapolated to the entrance of the muon spectrometer. Similarly, mu_pt_exBI is the transverse momentum of the combined track at the BI chambers.

We also store the track parameters for the extrapolated MS track contained in the combined muon

Default name	Filled on	Description
mu_msextrap_pt	Event	extrapolated MS pt
mu_msextrap_d0	Event	extrapolated MS D0
mu_msextrap_z0	Event	extrapolated MS Z0
mu_msextrap_phi	Event	extrapolated MS Phi
mu_msextrap_eta	Event	extrapolated MS Eta

The diagonal errors:

Default name	Filled on	Description
mu_covqoverp	Event	fit error in q/p
mu_covtheta	Event	error in theta
mu_covphi	Event	error in phi

mu_covd0	Event	error in d0
mu_covz0	Event	error in z0

- The errors above are also stored for ms and id tracks. In each case, the stored errors are relative to the default perigee.

The track's charge and fit chi2 with degrees of freedom are always stored (combined, muon spectrometer and inner detector)

Default name	Filled on	Description
mu_charge	Event	track charge
mu_fitchi2	Event	chi2 of track fit
mu_fitndof	Event	fit degrees of freedom

Hit-on-track summary

Default name	Filled on	Description
mu_hasBM	Event	Whether the track has a measurement in the BI station
mu_hasBM	Event	Whether the track has a measurement in the BM station
mu_hasBM	Event	Whether the track has a measurement in the BO station
mu_hasEI	Event	Whether the track has a measurement in the EI station
mu_hasEM	Event	Whether the track has a measurement in the EM station
mu_hasEO	Event	Whether the track has a measurement in the EO station
mu_hasRPC1Eta	Event	Whether the track has an RPC eta hit in layer 1 (BM)
mu_hasRPC2Eta	Event	Whether the track has an RPC eta hit in layer 2 (BM)
mu_hasRPC3Eta	Event	Whether the track has an RPC eta hit in layer 3 (BO)
mu_hasRPC1Phi	Event	Whether the track has an RPC phi hit in layer 1 (BM)
mu_hasRPC2Phi	Event	Whether the track has an RPC phi hit in layer 2 (BM)
mu_hasRPC3Phi	Event	Whether the track has an RPC phi hit in layer 3 (BO)
mu_barrelSectors	Event	Number of barrel (phi) sectors in which the track has MDT hits
mu_endcapSectors	Event	Number of endcap (phi) sectors in which the track has MDT hits
mu_nBLHits	Event	Number of hits in the pixel B-layer
mu_nPixHits	Event	Number of hits in the pixels
mu_nSCTHits	Event	Number of hits in the SCT
mu_nTRTHits	Event	Number of hits in the TRT
mu_nTRTHighTHits	Event	Number of hits in the TRT passing high-threshold
mu_nBLSharedHits	Event	Number of shared hits in the pixel B-layer
mu_nPixSharedHits	Event	Number of shared hits in the pixels
mu_nSCTSharedHits	Event	Number of shared hits in the SCT
mu_nPixHoles	Event	Number of pixel layers on track with hit absence
mu_nSCTHoles	Event	Number of SCT holes
mu_nTRTOutliers	Event	Number of TRT outliers
mu_nTRTHighTOutliers	Event	Number TRT high-threshold outliers
mu_nMDTHits	Event	Number of MDT hits on track
mu_nMDTHoles	Event	Number of MDT holes
mu_nRPCEtaHits	Event	Number of RPC eta hits on track
mu_nRPCEtaHoles	Event	Number of RPC eta holes
mu_nRPCPhiHits	Event	Number of RPC phi hits on track
mu_nRPCPhiHoles	Event	Number of RPC phi holes
mu_nTGCEtaHits	Event	Number of TGC eta hits on track
mu_nTGCEtaHoles	Event	Number of TGC eta holes
mu_nTGCPhiHits	Event	Number of TGC phi hits on track

mu_nTGCPhiHoles	Event	Number of TGC phi holes
mu_nCSCEtaHits	Event	Number of CSC eta hits on track
mu_nCSCEtaHoles	Event	Number of CSC eta holes
mu_nCSCPhiHits	Event	Number of CSC phi hits on track
mu_nCSCPhiHoles	Event	Number of CSC phi holes
mu_nOutliersOnTrack	Event	Number of measurements marked as outliers

- Again, the above variables are replicated for MS and ID tracks as well, hence mu_ms_nRPCPhiHits or mu_id_nPixHits
- ID technology hit summary variables exist only for combined and ID muons. So, mu_ms_nPixHits does not exist, but mu_id_nPixHits and mu_nPixHits do.
- MS technology hit summary variables exist only for combined and MS muons. So, mu_id_nTGCEtaHits does not exist, but mu_ms_nTGCEtaHits and mu_nTGCEtaHits do.

Hit-on-track details

Default name	Filled on	Description
mu_hitResidual	Event	hit residual w.r.t. track
mu_hitError	Event	hit error
mu_hitTech	Event	0/1/2/3 for MDT/CSC/TGC/RPC
mu_hitId	Event	MuonFixedId for the hit
mu_hitX	Event	Global position, x
mu_hitY	Event	Global position, y
mu_hitZ	Event	Global position, z
mu_hitR	Event	Hit radius for MDT
mu_hitT	Event	Hit drift time for MDT

- Individual track hits are stored only for the combined track. This behavior can be modified by changing (compile-time) flags.

Running over multiple track collections

By default, the MuonDumper filler runs once, for a combined muon collection. However, it is also possible to run it multiple times over track collections, and over multiple muon collections. This is useful, for example, to perform comparative studies between Muonboy and Moore tracks, or MS and ID tracks, or Staco and Muid etc. TrackCollection mode can be toggled on in the job obtions by setting **DoAnalysisMuon** to false. (the requisite configurations are included in the joboptions file NtupleMaker_ESD.py.)

In this mode, the relevant track parameters, extrapolations, hit summaries and details above are dumped depending on the track kind (ms or id), specified using the option **TrackType**.

Default name	Filled on	Description
mu_tag_pt	Event	Transverse momentum (GeV)
mu_tag_charge	Event	Charge
mu_tag_eta	Event	Eta
mu_tag_phi	Event	Azimuthal angle
mu_tag_chi2ndof	Event	chi2 per degree of freedom
mu_tag_d0	Event	Transverse impact parameter
mu_tag_z0	Event	Longitudinal impact parameter

Muon Segments

Default name	Filled on	Description
seg_nSeg	Event	Number of muon segments in event
seg_X	Event	Segment global X position
seg_Y	Event	Segment global Y position
seg_Z	Event	Segment global Z position
seg_phi	Event	Segment global phi direction
seg_theta	Event	Segment global theta direction
seg_locX	Event	Segment local X position
seg_locY	Event	Segment local Y position
seg_locAngleXZ	Event	Segment local XZ angle
seg_locAngleYZ	Event	Segment local YZ angle
seg_chi2	Event	Segment fit chi2
seg_ndof	Event	Segment fit degrees of freedom
seg_sector	Event	Segment phi sector 1-16 (all technologies except TGC)
seg_nHits	Event	Number of hits on segment
seg_hitResidual	Event	hit residual w.r.t. segment
seg_hitError	Event	hit error
seg_hitTech	Event	0/1/2/3 for MDT/CSC/TGC/RPC
seg_hitId	Event	MuonFixedId for the hit
seg_hitX	Event	Global position, x
seg_hitY	Event	Global position, y
seg_hitZ	Event	Global position, z
seg_hitR	Event	Hit radius for MDT
seg_hitT	Event	Hit drift time for MDT

Raw hits for the muon spectrometer

MDT INFO

Default name	Filled on	Description
nMdtPRD	Event	Number of MDT PRDs in Event
MDT_fixedId	Event	Fixed Identifier of PRD
MDT_x	Event	Global x Position of MDT Prd
MDT_y	Event	Global y Position of MDT Prd
MDT_tdc	Event	TDC of MDT Prd
MDT_adc	Event	ADC of MDT Prd
MDT_driftRadius	Event	Drift Radius of MDT Prd
MDT_driftRadius Error	Event	Drift Radius error of MDT Prd
MDT_status	Event	Status of PRD

RPC INFO

Default name	Filled on	Description
nRpcPRD	Event	Number of Rpc PRDs in Event
RPC_x	Event	Global x Position of RPC Prd
RPC_y	Event	Global y Position of RPC Prd
RPC_time	Event	RPC time
RPC_trigInfo	Event	RPC trigInfo
RPC_ambiguityFlag	Event	RPC ambiguity flag

RPC_inRibs	Event	in Ribs ?
RPC_doubletR	Event	R doublet of RPC
RPC_doubletZ	Event	Z doublet of RPC
RPC_stripWidth	Event	strip Width
RPC_stripLength	Event	strip Length
RPC_stripPitch	Event	strip Pitch
RPC_station	Event	Rpc Station
RPC_measuresPhi	Event	Measures Phi

TGC INFO

Default name	Filled on	Description
nTgcPRD	Event	Number of TGC PRDs in Event
TGC_x	Event	Global x Position of TGC Prd
TGC_y	Event	Global y Position of TGC Prd
TGC_z	Event	Global y Position of TGC Prd
TGC_shortWidth	Event	short Width of TGC strip
TGC_longWidth	Event	long Width of TGC strip
TGC_isStrip	Event	is strip or wire
TGC_channel	Event	channel no
TGC_eta	Event	eta of Station
TGC_phi	Event	phi of Station
TGC_bunch	Event	bunch (1 - previous, 2 - current , 3 - next one)

CSC INFO

Default name	Filled on	Description
nCscPRD	Event	Number of CSC PRDs in Event
CSC_x	Event	Global x Position of CSC Prd
CSC_y	Event	Global y Position of CSC Prd
CSC_z	Event	Global y Position of CSC Prd
CSC_time	Event	CSC time
CSC_charge	Event	CSC charge
CSC_status	Event	CSC status

| CSC_station | Event | CSC station

CSC_charge	Event	vector of sampling charges
CSC_samplingPhase	Event	Sampling phase
CSC_measuresPhi	Event	Measures Phi
CSC_strip	Event	is Strip

Inner Detector Tracks

Information about inner detector tracks is dumped by TrackDumper (src, header)

Default name	Filled on	Description
track_ntracks	Event	Number of ID Tracks in event
track_index	Event	Index of track for linking to other algorithms
track_qoP	Event	Track charge divided by momentum (at 0,0)
track_phi	Event	Track phi (at 0,0)
track_theta	Event	Track theta (at 0,0)
track_d0	Event	Track d0 (wrt 0,0)

track_zo	Event	Track z0 (wrt 0,0)
track_corr_qoP	Event	Track charge divided by momentum (at primary vertex)
track_corr_phi	Event	Track phi (at primary vertex)
track_corr_theta	Event	Track theta (at primary vertex)
track_corr_d0	Event	Track d0 (wrt primary vertex)
track_corr_zo	Event	Track z0 (wrt primary vertex)
track_caloLossEnergy	Event	Energy in calorimeter along track
track_caloLossEnergyLoose	Event	Energy (no calo cell cut) in calorimeter along track
track_caloEnergyInCone	Event	Energy in calorimeter in cone around track
track_caloEnergyInCone	Event	Energy (no calo cell cut) in calorimeter in cone around track
track_vtx_x	Event	Track vertex x position
track_vtx_y	Event	Track vertex y position
track_vtx_z	Event	Track vertex z position
track_trthits	Event	TRT hits on track
track_scthits	Event	SCT hits on track
track_pixelhits	Event	Pixel hits on track
track_blayerhits	Event	B-layer hits on track
track_fixX2OverNDOF	Event	X^2 over number of degrees of freedom for track fit
sct_barrel_bias	Block	SCT barrel bias voltage
sct_endcap_abias	Block	SCT endcap a bias voltage
sct_endcap_cbias	Block	SCT endcap c bias voltage
track_cell_trkIndex	Event	Index of track calo cell is associated with
track_cell_eta	Event	CaloCell eta
track_cell_eta	Event	CaloCell phi
track_cell_energy	Event	CaloCell energy
track_cell_layer	Event	CaloCell layer in calorimeter
track_cell_isOnTrack	Event	Whether or not the extrapolated track goes through the cell
track_cell_isLoose	Event	Whether or not the calo cell passes noise suppression

Trigger

Information about the trigger is dumped by the TriggerDumper (src, header)

Default name	Filled on	Description
trig_triggers	Run	Name of each trigger chain in the configuration
trig_prescales	Block	Prescale of each trigger chain in the configuration
trig_bits	Event	Trig decision bits for each trigger chain in the configuration
trig_mbts_times	Event	Timing of MBTS cell
trig_mbts_energy	Event	Energy of MBTS cell
trig_mbts_quality	Event	Quality of MBTS cell
trig_mbts_eta	Event	Sign of eta of MBTS cell
trig_mbts_phi	Event	Phi of MBTS cell
trig_mbts_channel	Event	Channel ID of MBTS cell
trig_calo_ncellA	Event	Number of calo cells on endcap A above noise threshold
trig_calo_ncellC	Event	Number of calo cells on endcap C above noise threshold
trig_calo_timeA	Event	Energy weighted average time of calo cells in endcap A above noise threshold
trig_calo_timeC	Event	Energy weighted average time of calo cells in endcap C above noise threshold
trig_calo_energyA	Event	Total energy of calo cells in endcap A above noise threshold
trig_calo_energyC	Event	Total energy of calo cells in endcap C above noise threshold

Vertex

Information is provided about two types of vertices: The primary vertex is filled from VxDumper (src, header) and displaced verticies are filled from V0Dumper (src, header)

Primary Vertex

Default name	Filled on	Description
vx_id	Event	ID of primary vertex for connecting to tracks
vx_x	Event	X position of primary vertex
vx_y	Event	Y position of primary vertex
vx_z	Event	Z position of primary vertex
vx_ntrks	Event	Number of tracks from this vertex
vxtrk_v0id	Event	ID of vertex this track belongs to
vxtrk_trkid	Event	ID of track to link to

Displaced Verticies

Default name	Filled on	Description
v0_id	Event	ID of displaced vertex for connecting to tracks
v0_x	Event	X position of displaced vertex
v0_y	Event	Y position of displaced vertex
v0_z	Event	Z position of displaced vertex
v0_ntrks	Event	Number of tracks from this vertex
v0trk_v0id	Event	ID of vertex this track belongs to
v0trk_trkid	Event	ID of track to link to

Development

The section contains information about development plans for the framework.

Support

This section contains information about getting help with using the HarvardAnalysisFramework. The first section contains answers to frequently asked questions, while the next section contains information about who you should contact if you need help or want to offer suggestions.

Frequently Asked Questions

None yet!

Where to go for help

The best way to get help is to contact the people in charge of the code you are interested in. They are listed below:

Role	Contact information
Primary contact, framework development, urgent requests/bugs	BenSmith
Secondary contact if above is not responding	CorrinneMills , KevinBlack
Muon and Muon segment	SrivasPrasad, KevinBlack, AlbertoBelloni, VerenaMartinez
Electron and photon	CorrinneMills
Trigger	BenSmith
ID Tracking	BenSmith, CorrinneMills
Calorimeter	LauraJeanty
MC Truth	MichaelKagan
Jets	GiovanniZeviDellaPorta

-- BenSmith - 11-Feb-2010

This topic: Main > HarvardAnalysisFramework
Topic revision: r16 - 2010-04-16 - SrivasPrasad