JUNO Event Data Model and Tutorial

Xingtao Huang Jiaheng Zou Tao Lin Weidong Li

on behalf of offline software group

2018.05.13. Wuhan University

Goals

- Brief Introduction to Event Data Model
- Examples on How to use Event Data Model
  - Define Event Data Model Class
  - Retrieve event data from input file
  - Create and write out EDM objects
  - Use RootWriter to write plain TTrees/Tntuples
  - Perform Event Data Navigation
- Summary

Preparation

All 5 examples will use the several root files

Setup Environments

Set up Environments

$ source /afs/ihep.ac.cn/soft/juno/JUNO-ALL-SLC6/Release/J17v1r1/setup.sh (IHEP)

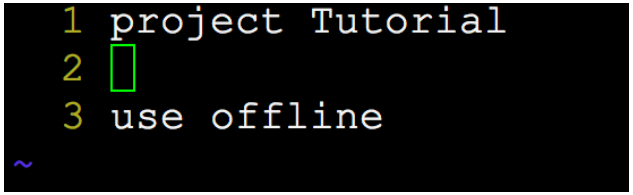
$ source ~/juno-dev/setup.sh (Virtual Machine)

Create a new project

$ cmt create_project Tutorial

$ cd Tutorial/cmt/

$ vim project.cmt



$ cd ../

$ export CMTPROJECTPATH=$PWD:$CMTPROJECTPATH

Check out packages

Check out Example for definition of DummyHeader and DummyEvent

$ svn co http://juno.ihep.ac.cn/svn/juno/people/zoujh/example/SecondToy SecondToy

Check out example on how to use Event Data

$ svn co http://juno.ihep.ac.cn/svn/juno/people/huangxt/EDMTutorial EDMTutorial

Check the packages

$ cd Tutorial/

$ ls

```
[huangxt@lxslc610 Tutorial]$ ls
cmt  EDMTutorial  InstallArea  SecondToy
```

Generate data sample files

$ cd EDMTutorial/cmt/
$ cmt config
$ cmt make
$ source setup.sh
$ cd ../share/

```
[huangxt@lxslc607 share]$ ls
preparefile.sh  runCA.py  run.py  runRW.py
```

preparefile.sh: run generator, detsim, electronics, calib and rec algorithms and save EDM data into root files.

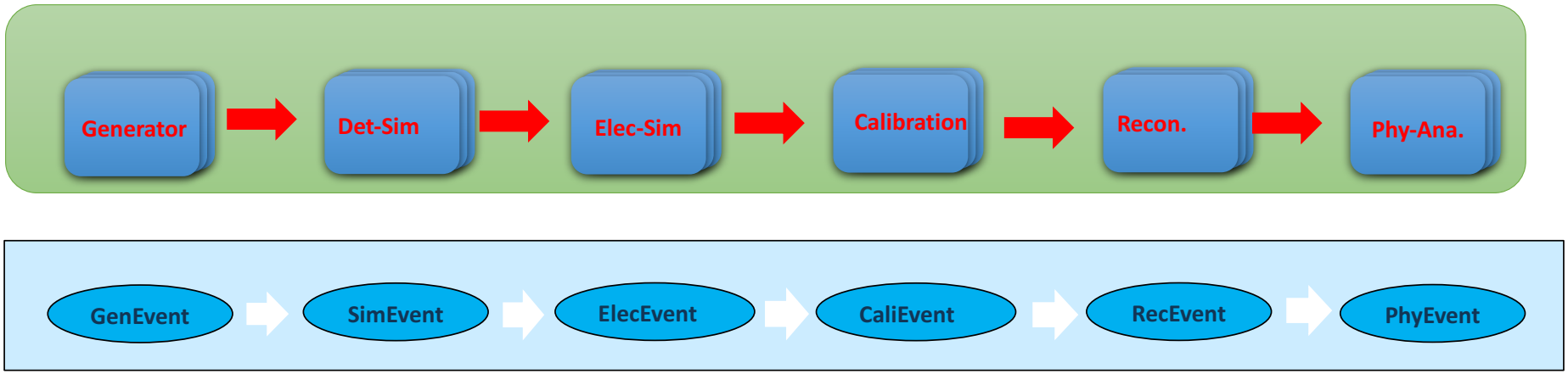
```
1 #/bin/bash
2 python $TUTORIALROOT/share/tut_detsim.py  --evtmax 100 gun
3 python $TUTORIALROOT/share/tut_det2calib.py --evtmax 100
4 python $TUTORIALROOT/share/tut_calib2rec.py --evtmax 100
```

$source preparefile.sh

Brief Introduction to Event Data Model

Offline Data Processing & Event Data Flow

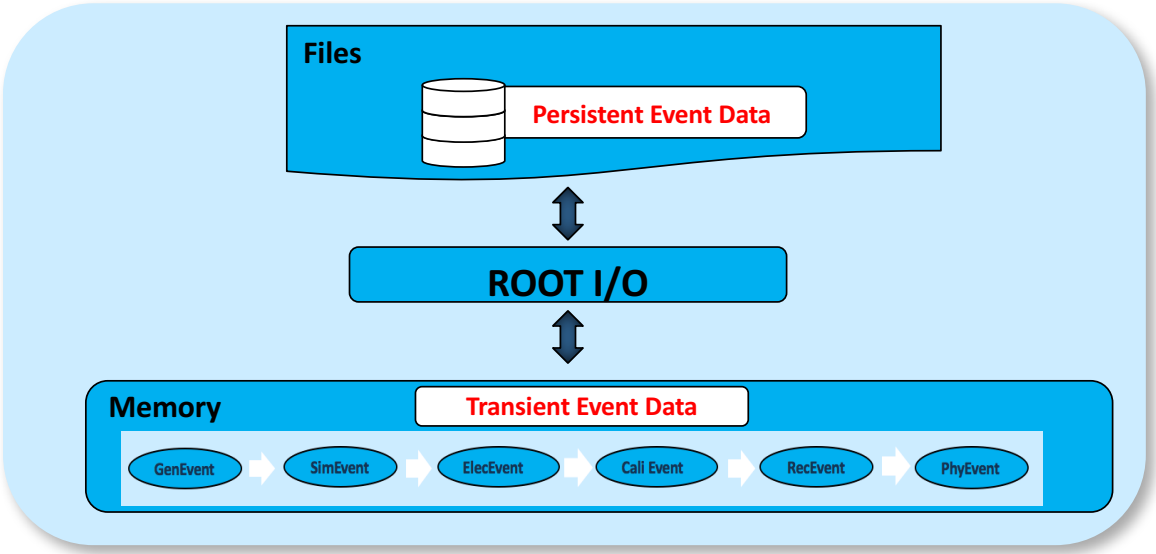
- Offline data processing



- Each processing stage has its specific event data
- Event Data Model defines event information at each stage and their correlations between different stages

Event Data: Transient and Persistent

- Two types of Event Data
  - Transient Event Data
    - **In-memory data or data during processing**
    - **Managed by framework**
  - Persistent Event Data
    - **Data on disk**
    - **ROOT format**
- Typically Objects of C++ Classes
- Same definition for both types of Event Data
- Root I/O are implemented with Services, which are configurable
  - RootInputSvc
  - RootOutputSvc

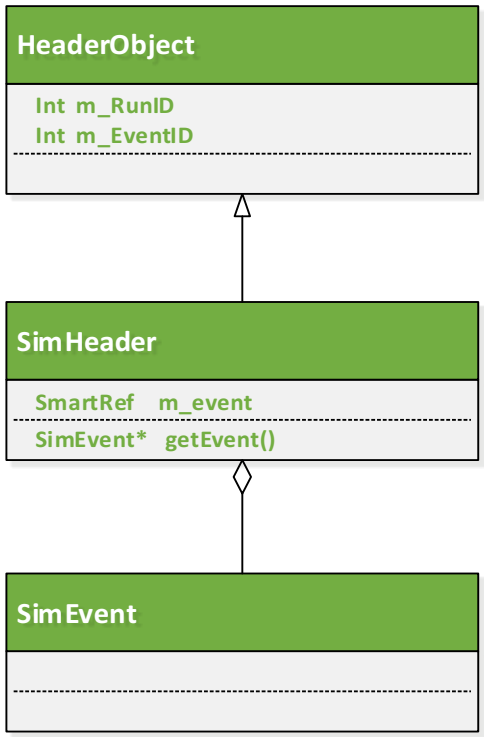


More information can be found in
Chinese Physics C, 41(6): 066201 (2017)

Design of Event Data Model

- **Two-layer Structure Event Data for each Stage (See Example 1)**
  - Header Object
    - XxxHeader: tag information, fast event filtering without reading the whole data object into memory
  - Event Object
    - XxxEvent: event data for detailed analysis
  - They are related with SmartRef, liking a smart C++ pointer
    - Easily get the event with the header via getEvent() interface

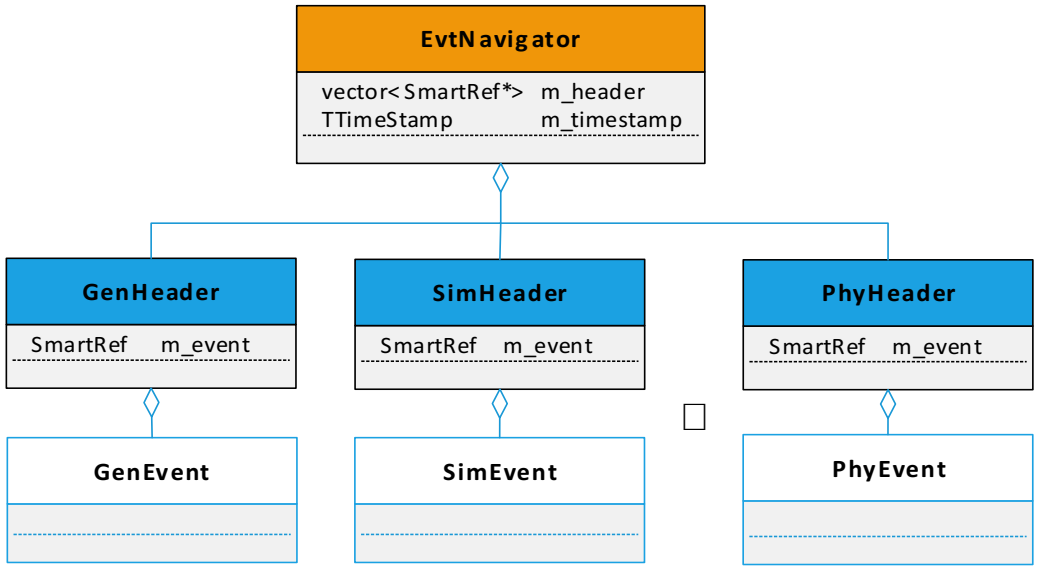
More information can be found in
Chinese Physics C, 41(6): 066201 (2017)



Design of Event Data Model

- **EvtNavigator is developed for correlation between different stages** (See Example 5)

  - Serve as index of all event objects
  - Provide convenient navigations between the event data at different data processing stages



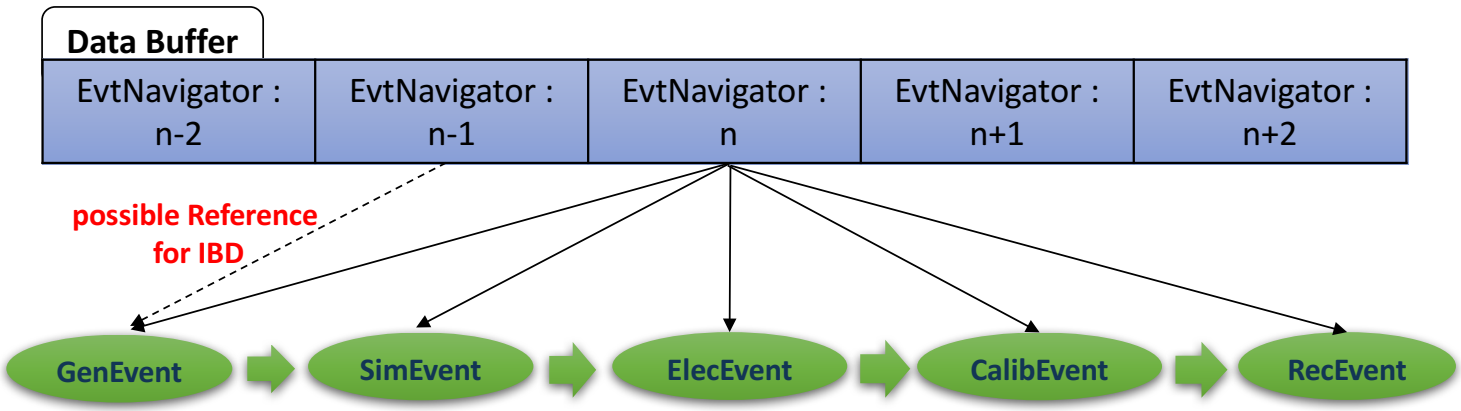
Use cases of EvtNavigator

- Build the correlation of event data at different stages

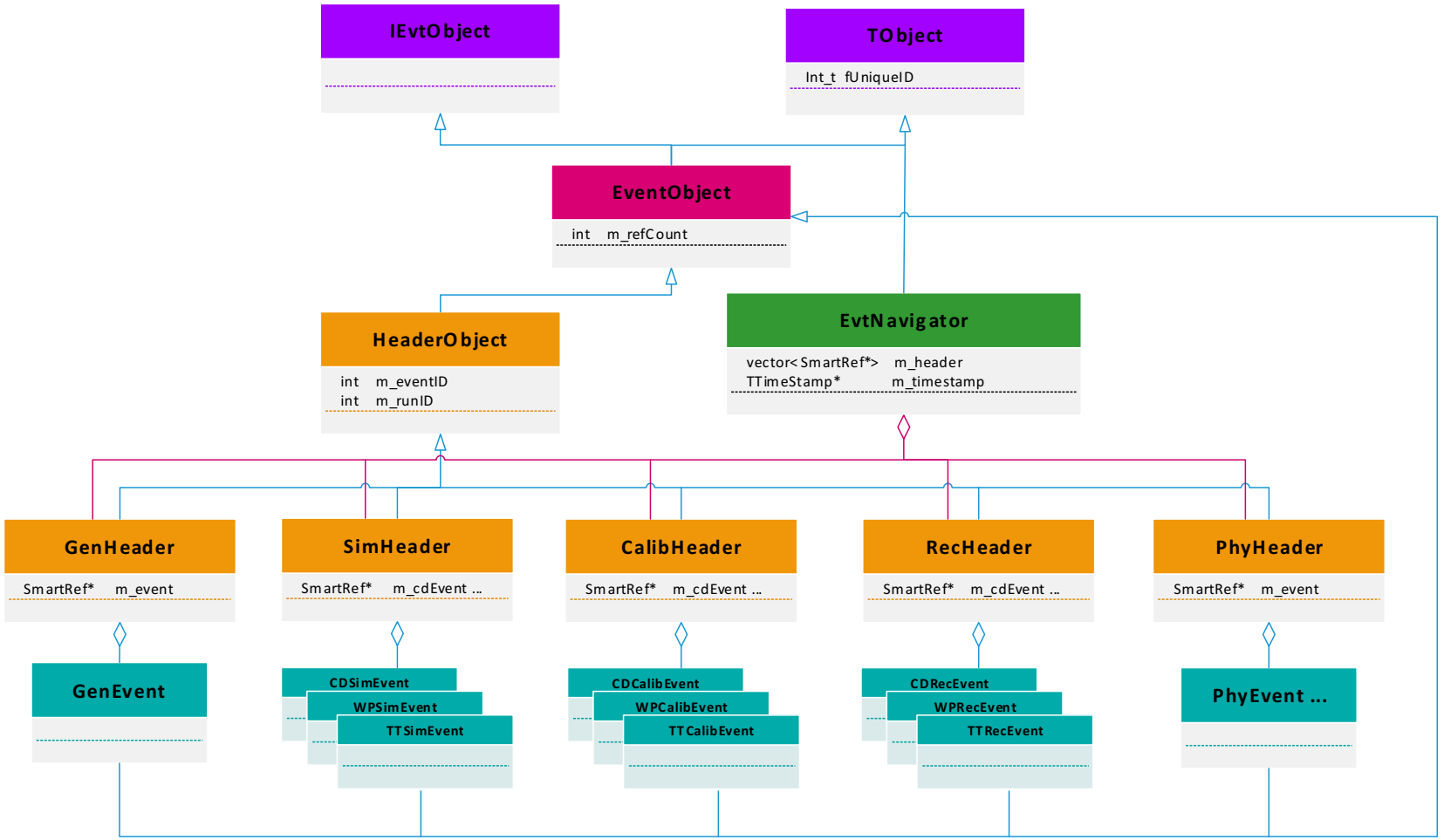
  Help users to do the correlational analysis, for example: navigate from reconstructed event to MC truth

- Describe IBD signals accurately

  The prompt-delayed readouts pair of an IBD reaction actually come from one event at the generator or detector simulation stages.



Layout of JUNO Event Data Model



Implementation of Event Data Model

- TObject

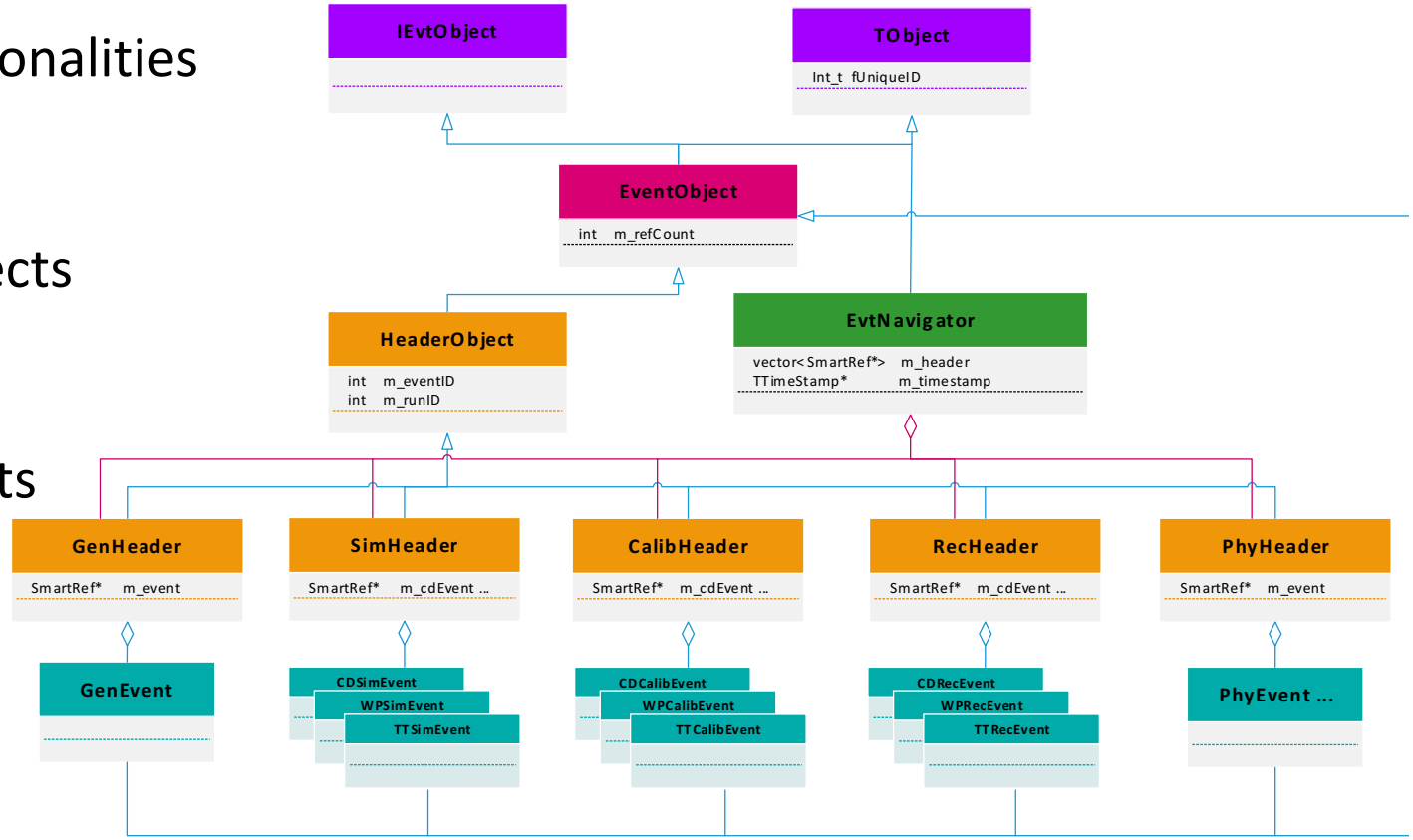
  Make use of ROOT functionalities

- HeaderObject

  Base class of header objects

- EventObject

  Base class of Event objects



Current EDM for different stages

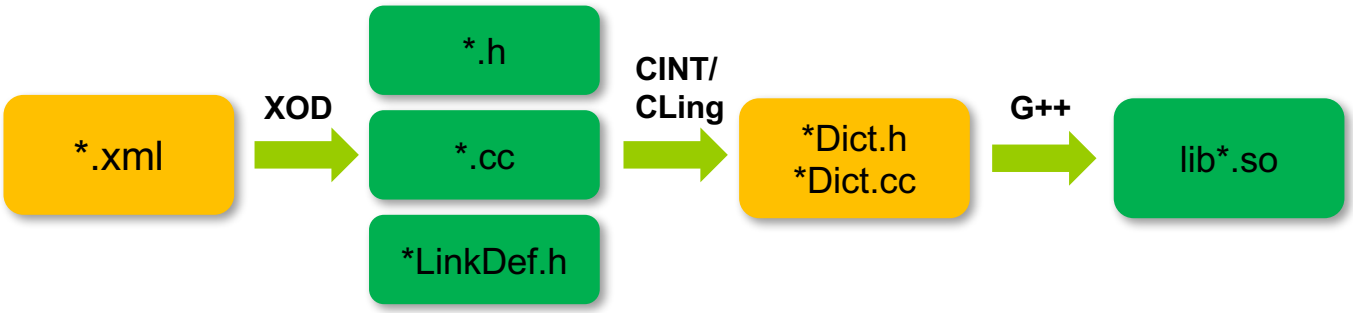
- GenEvent: information of physics generator
  offline/DataModel/GenEventV2
- SimEvent: detector simulation
  offline/DataModel/SimEventV2
- CalibEvent: waveform reconstruction
  offline/DataModel/CalibEvent
- ElecEvent: readout of electronic simulation
  offline/DataModel/ElecEvent
- RecEvent: energy/vertex reconstruction
  offline/DataModel/RecEvent
- RecTrackEvent: track reconstruction
  offline/DataModel/RecEvent
- Detailed information is attached on backup slides, and docdb:
  http://juno.ihep.ac.cn/cgi-bin/Dev_DocDB/ShowDocument?docid=2052

Example 1:

Define New Event Data Model Classes with XOD

How to define a New Event Data Model classes

- **Traditionally writing C++ Code manually**
  - Many repeatable work such as Getters and Setters
  - Difficult to be maintained
- **Use XML file to define EDM**
  - Strong syntax (DTD, XML Schema)
  - More readable, easier to maintain
  - Automatically generate the Get-, Set-functions, ROOT I/O Streamers
- **A tool, XmlObjDesc (XOD), is developed to automatically generate class codes**



Example 1: define DummyHeader and DummyEvent

```
1  <?xml version="1.0" encoding="UTF-8" ?>
2  <!DOCTYPE xdd SYSTEM "xdd.dtd">
3
4  <xdd>
5      <package name="DummyEvent">
6
7          <import name="Event/HeaderObject"/>
8          <import name="Event/DummyEvent"/>
9
10         <class name="DummyHeader"
11                author="LI Teng"
12                desc="Dummy Header Class">
13
14             <base name="HeaderObject"/>
15
16             <SmartRelation  type="JM::DummyEvent"
17                             name="event"
18                             desc="SmartRef to the DummyEvent"
19                             nonconstaccessor="TRUE"/>
20
21         </class>
22     </package>
23 </xdd>
```

◆DummyHeader.xml

- Line 7-8: include headers
- Line 14: Base class
- Line 16-19: Smart reference to the DummyEvent object

```
1  <?xml version="1.0" encoding="UTF-8" ?>
2  <!DOCTYPE xdd SYSTEM "xdd.dtd">
3
4  <xdd>
5      <package name="DummyEvent">
6
7          <import name="Event/EventObject"/>
8
9          <class name="DummyEvent"
10                author="LI Teng"
11                desc="Dummy Event Class">
12
13             <base name="EventObject"/>
14
15             <attribute name="energy"
16                        type="double"
17                        desc="energy of the event"
18                        init="0.0"/>
19
```

◆DummyEvent.xml (part)

- Line 15-18: Define attributes (member variables)

Run the job

Look at xml file in xml/

$ cd SecondToy/DummyEvent/

$ cd cmt/

$ cmt config

$ cmt make

Look at automatically generated header file and event file in Event/ and src/

$ vim ../Event/DummyHeader.h

$ vim ../src/DummyHeader.cc

$ vim ../Event/DummyEvent.h

$ vim ../src/DummyEvent.cc

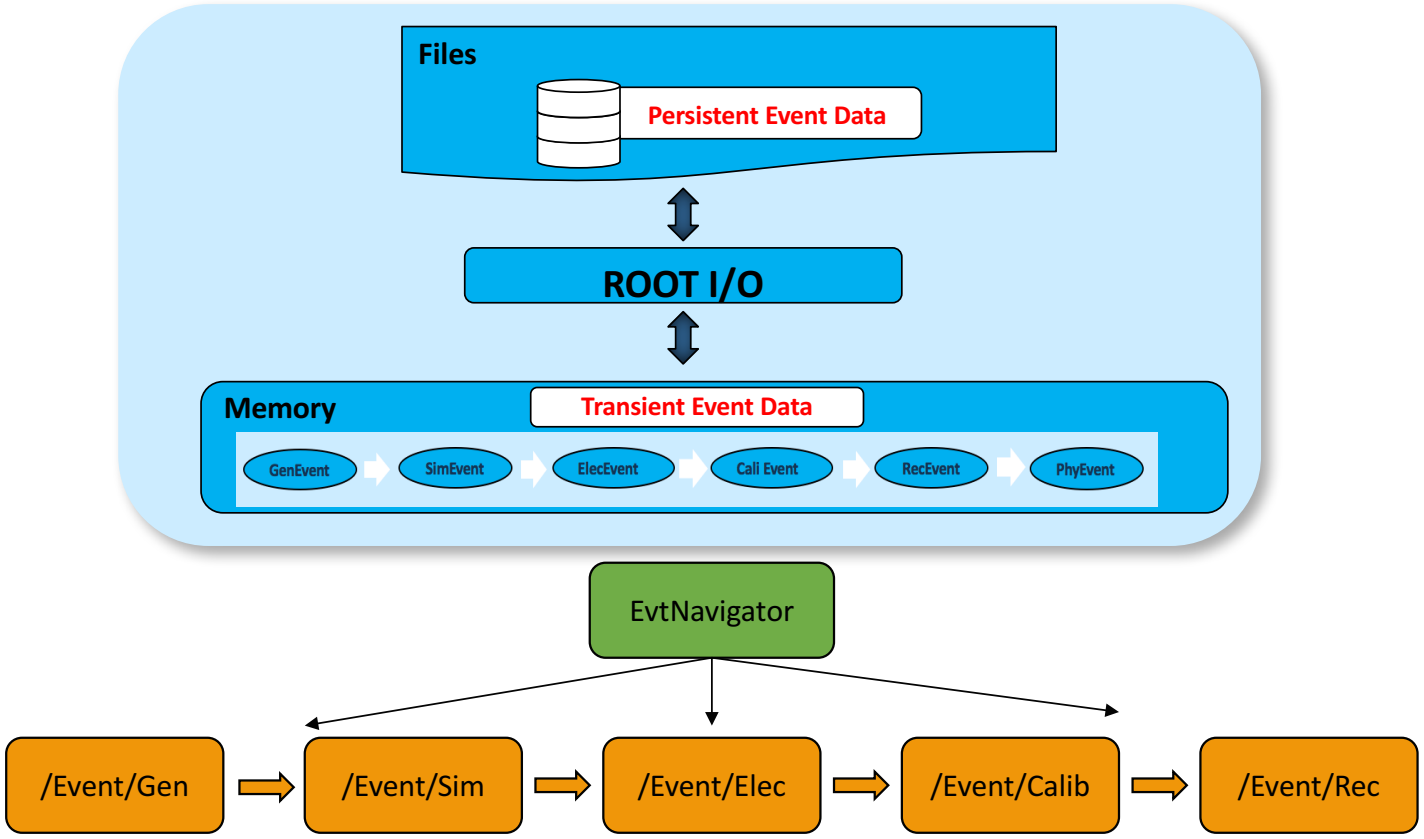
Example 2:

Retrieve Event Data from the Buffer with the path

Example 3:

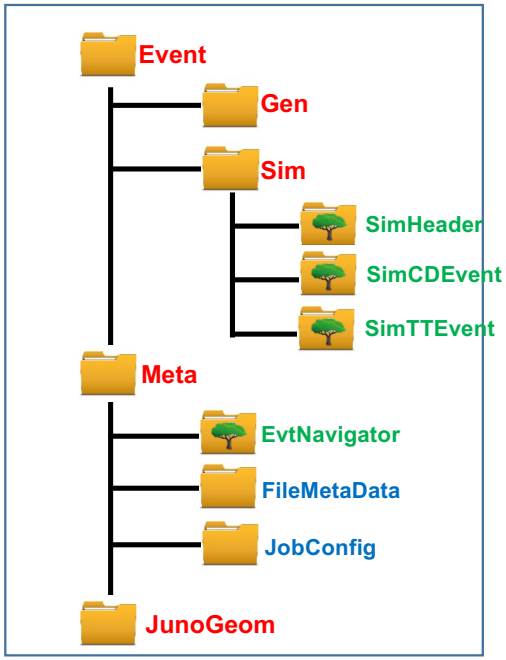
Create Event Data and put it into the Buffer with the path

Event Data Structure in Memory and Files



Event Data Structure in Memory

ROOT File Structure



Example 2:Retrieve Event Data with the path

Tutorial/EDMTutorial

- src/TutAlg.h
- src/TutAlg.cc

- Line 2-7
  - include headers of DataBuffer and EDM
- Line 17-22
  - Get the DataBuffer

```
 1 #include "TutAlg.h"
 2 #include "SniperKernel/AlgFactory.h"
 3 #include "BufferMemMgr/IDataMemMgr.h"
 4 #include "BufferMemMgr/EvtDataPtr.h"
 5 #include "EvtNavigator/EvtNavigator.h"
 6 #include "Event/SimHeader.h"
 7 #include "Event/CalibHeader.h"
 8
 9 DECLARE_ALGORITHM(TutAlg);
10
11 TutAlg::TutAlg(const std::string& name)
12     : AlgBase(name)
13 {}
14
15 bool TutAlg::initialize()
16 {
17     SniperDataPtr<JM::NavBuffer>  navBuf(getScope(), "/Event");
18     if ( navBuf.invalid() ) {
19         LogError << "cannot get the NavBuffer @ /Event" << std::endl;
20         return false;
21     }
22     m_buf = navBuf.data();
23     return true;
24 }
```

Example 2:Retrieve Event Data with the path

- Line 28
  - Get the EvtNavigator
- Line 29
  - One way to get the SimHeader
- Line 38-29
  - Second way to get the SimHeader
- Line 31 / Line 41
  - get the SimEvent

```
26 bool TutAlg::execute()
27 {
28     JM::EvtNavigator* nav = m_buf->curEvt();
29     JM::SimHeader* header = (JM::SimHeader*)nav->getHeader("/Event/Sim");
30     if (header) {
31         JM::SimEvent* event = (JM::SimEvent*)header->event();
32         // Get the event data
33     } else {
34       LogError << "cannot get the header @/Event/Sim" <<std::endl;
35       return false;
36     }
37
38     EvtDataPtr<JM::SimHeader> edp(getScope(), "/Event/Sim");
39     JM::SimHeader* Header = edp.data();
40     if (Header) {
41         JM::SimEvent* Event = (JM::SimEvent*)Header->event();
42         // Get the event data
43     } else {
44         LogError << "cannot get the header @/Event/Sim" <<std::endl;
45         return false;
46     }
47
48     JM::CalibHeader* ch = new JM::CalibHeader;
49     JM::CalibEvent* ce = new JM::CalibEvent;
50     ch->setEvent(ce);
51     nav->addHeader("/Event/Calib", ch);
52
53     // Fill your data in CalibHeader and CalibEvent
54     return true;
55 }
56
```

All data will be read in from input files automatically for each execute().

Example 3:Create Event Data and Put in the Buffer

- Line 48
  - Create new CalibHeader
- Line 49
  - Create new CalibEvent
- Line 50
  - Set relationship between CalibHeader and CalibEvent
- Line 51
  - Put CalibHeader into the buffer with the path “/Event/Calib”

```
26 bool TutAlg::execute()
27 {
28     JM::EvtNavigator* nav = m_buf->curEvt();
29     JM::SimHeader* header = (JM::SimHeader*)nav->getHeader("/Event/Sim");
30     if (header) {
31         JM::SimEvent* event = (JM::SimEvent*)header->event();
32         // Get the event data
33     } else {
34       LogError << "cannot get the header @/Event/Sim" <<std::endl;
35       return false;
36     }
37
38     EvtDataPtr<JM::SimHeader> edp(getScope(), "/Event/Sim");
39     JM::SimHeader* Header = edp.data();
40     if (Header) {
41         JM::SimEvent* Event = (JM::SimEvent*)Header->event();
42         // Get the event data
43     } else {
44         LogError << "cannot get the header @/Event/Sim" <<std::endl;
45         return false;
46     }
47
48     JM::CalibHeader* ch = new JM::CalibHeader;
49     JM::CalibEvent* ce = new JM::CalibEvent;
50     ch->setEvent(ce);
51     nav->addHeader("/Event/Calib", ch);
52
53     // Fill your data in CalibHeader and CalibEvent
54     return true;
55 }
56
```

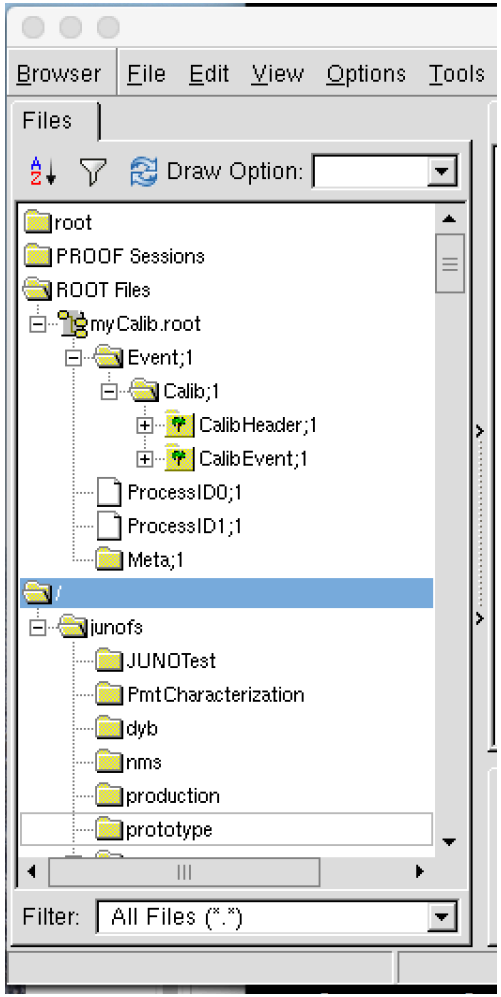
Python script: share/run.py

- Line 4:
  - Create the Sniper task
- Line 9:
  - Create the Algorithm
- Line 12:
  - Setup DataBuffer
- Line 14-16:
  - Setup input service
- Line 18-19:
  - Setup output service
  - Setup the output stream

```
 1 #!/usr/bin/env python
 2
 3 import Sniper
 4 task = Sniper.Task("task")
 5 task.asTop()
 6 task.setLogLevel(0)
 7
 8 import EDMTutorial
 9 alg = task.createAlg("TutAlg/alg")
10
11 import BufferMemMgr
12 bufMgr = task.createSvc("BufferMemMgr")
13
14 import RootIOSvc
15 riSvc = task.createSvc("RootInputSvc/InputSvc")
16 riSvc.property("InputFile").set(["sample_detsim.root"])
17
18 roSvc = task.createSvc("RootOutputSvc/OutputSvc")
19 roSvc.property("OutputStreams").set({"/Event/Calib": "myCalib.root"})
20
21 task.setEvtMax(10)
22 task.show()
23 task.run()
```

Run the Job

```
$ cd EDMTutorial/cmt/
$ cmt config
$ cmt make
$ source setup.sh
$ cd ../share
$ python run.py
```



A new root file (myCalib.root) is created and save CalibHeader and CalibEvent in separate Trees

Example 4:

Output flat TTree/TNtuple

Output flat TTree/TNtuple

EDMTutorial/src/RWTut.h
EDMTutorial/src/RWTut.cc

```
 9 class RWTut : public AlgBase
10 {
11     public :
12
13         RWTut(const std::string& nam
14         virtual ~RWTut();
15
16         virtual bool initialize();
17         virtual bool execute();
18         virtual bool finalize();
19
20     private :
21
22         int     m_iEvt;
23         int     m_iLeaf;
24         double  m_dLeaf1;
25         double  m_dLeaf2;
26         double  m_dLeaf3;
27
28         TTree*       m_tree;
29         TNtuple*      m_tuple;
30 };
31
32 #endif
```

```
19 bool RWTut::initialize()
20 {
21     SniperPtr<RootWriter> m_rw("RootWriter");
22     if ( ! m_rw.valid() ) {
23         LogError << "Failed to get RootWriter instance!" << std::endl;
24         return false;
25     }
26
27     m_tree = new TTree("tree", "Simple tree");
28     m_tree->Branch("iEvt",  &m_iEvt,  "iEvt/I");
29     m_tree->Branch("iLeaf", &m_iLeaf, "iLeaf/I");
30
31     m_tuple = new TNtuple("ntuple","Simple ntuple","dLeaf1:dLeaf2:dLeaf3");
32
33
34     m_rw->attach("FILE1", m_tree);
35     m_rw->attach("FILE2", m_tuple);
36
37     LogInfo << " initialized successfully" << std::endl;
38
39     return true;
40 }
41
```

- Line 21
  - Get RootWriter
- Line 27-35
  - Create TTree/TNtuple, and attach them to output files

Output flat TTree/TNtuple

- Line 47-50
  - Set values of leaves
- Line 53
  - Fill the Tree
- Line 54
  - Fill Tuple

```
42 bool RWTut::execute()
43 {
44     ++m_iEvt;
45
46     //calculations
47     m_iLeaf = m_iEvt % 100;
48     m_dLeaf1 = (m_iEvt / 3.377111) * 0.019222;
49     m_dLeaf2 = (m_iEvt / 2.242222) * 0.031222;
50     m_dLeaf3 = (m_iEvt / 1.325111) * 0.045222;
51
52     //fill trees and ntuples
53     m_tree->Fill();
54     m_tuple->Fill(m_dLeaf1, m_dLeaf2, m_dLeaf3);
55
56     return true;
57 }
```

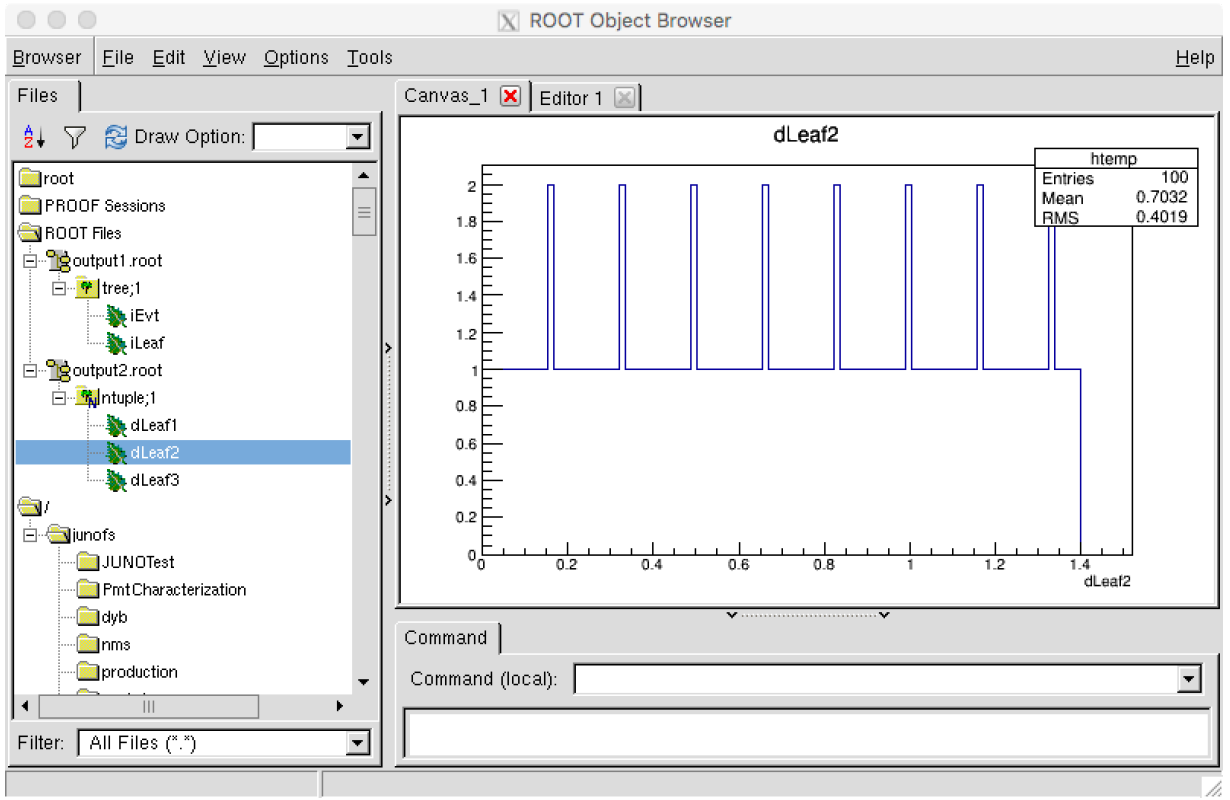
Python script: share/runRW.py

- Line 10-11
  - Get RootWriter
- Line 12
  - Set output files

```
 1 #!/usr/bin/env python
 2
 3 import Sniper
 4
 5 task = Sniper.Task("task")
 6 task.asTop()
 7 task.setLogLevel(0)
 8
 9 import RootWriter
10 task.property("svcs").append("RootWriter")
11 rw = task.find("RootWriter")
12 rw.property("Output").set({"FILE1": "output1.root", "FILE2": "output2.root"})
13
14 import EDMTutorial
15 alg = task.createAlg("RWTut")
16
17 task.setEvtMax(100)
18 task.show()
19 task.run()
~
```

Run the Job and Look at root files

$ python run.py



$ root.exe output1.root output2.root

Example 5:

Event Data Navigation

Event Data Navigation

- Use EvtNavigator to get information at different stages of an event
  - SimEvent
  - RecEvent
- Similar requirements with Correlation Analysis

EDMTutorial/src/CorAnaAlg .h
EDMTutorial/src/CorAnaAlg.cc

```
 2 #define COR_ANA_ALG_H
 3
 4 #include "SniperKernel/AlgBase.h"
 5 #include "EvtNavigator/NavBuffer.h"
 6
 7 class TTree;
 8
 9 class CorAnaAlg : public AlgBase
10 {
11     public :
12
13         CorAnaAlg(const std::string& name);
14
15         bool initialize();
16         bool execute();
17         bool finalize();
18
19     private :
20
21         int m_iEvt;
22         JM::NavBuffer* m_buf;
23         TTree* m_tree;
24
25         int m_nhits;
26         double m_energy;
27
28 };
29
30 #endif
```

Event Data Navigation

- Line 23-28
  - Get the DataBuffer
- Line 30-34
  - Get the RootWriter
- Line 36-41
  - Define the TTree

```
19 bool CorAnaAlg::initialize()
20 {
21     LogDebug << "initializing" << std::endl;
22
23     SniperDataPtr<JM::NavBuffer>  navBuf(getScope(), "/Event");
24     if ( navBuf.invalid() ) {
25         LogError << "cannot get the NavBuffer @ /Event" << std::endl;
26         return false;
27     }
28     m_buf = navBuf.data();
29
30     SniperPtr<RootWriter> rw("RootWriter");
31     if ( ! rw.valid() ) {
32         LogError << "Failed to get RootWriter instance!" << std::endl;
33         return false;
34     }
35
36     m_tree = new TTree("tree", "Simple tree");
37     m_tree->Branch("nhits",  &m_nhits,  "nhits/I");
38     m_tree->Branch("energy", &m_energy, "energy/D");
39
40     rw->attach("FILE1", m_tree);
41
42     return true;
43 }
```

Event Data Navigation

- Line 50
  - Get the EvtNavigator from DataBuffer
- Line 51-58
  - Get SimEvent and RecEvent from EvtNavigator
- Line 60-63
  - Fill the variables

```
45 bool CorAnaAlg::execute()
46 {
47     LogDebug << "executing: " << m_iEvt++ << std::endl;
48 
49     JM::EvtNavigator* nav = m_buf->curEvt();
50     JM::RecHeader* recheader = (JM::RecHeader*)(nav->getHeader("/Event/Rec"));
51     // Get the SimHeader after split
52     JM::SimHeader* simheader = (JM::SimHeader*)(nav->getHeader("/Event/Sim"));
53     if (!recheader || !simheader) {
54         LogError << "Failed to read in headers!" << std::endl;
55         return false;
56     }
57     JM::SimEvent* simevent = (JM::SimEvent*)simheader->event();
58     JM::CDRecEvent* cdrecevent = (JM::CDRecEvent*)recheader->cdEvent();
59 
60     m_nhits = simevent->getCDHitsVec().size();
61     m_energy = cdrecevent->energy();
62 
63     m_tree->Fill();
64 
65     return true;
66 }
```

Python script: share/runCA.py

- Line 20-21
  - Set up the input file list.
- Line 14-15
  - Set up the output file

When you want to correlational access events, data file at different stages should be added.

File at last stage serves as the master line. (sample_rec.root)

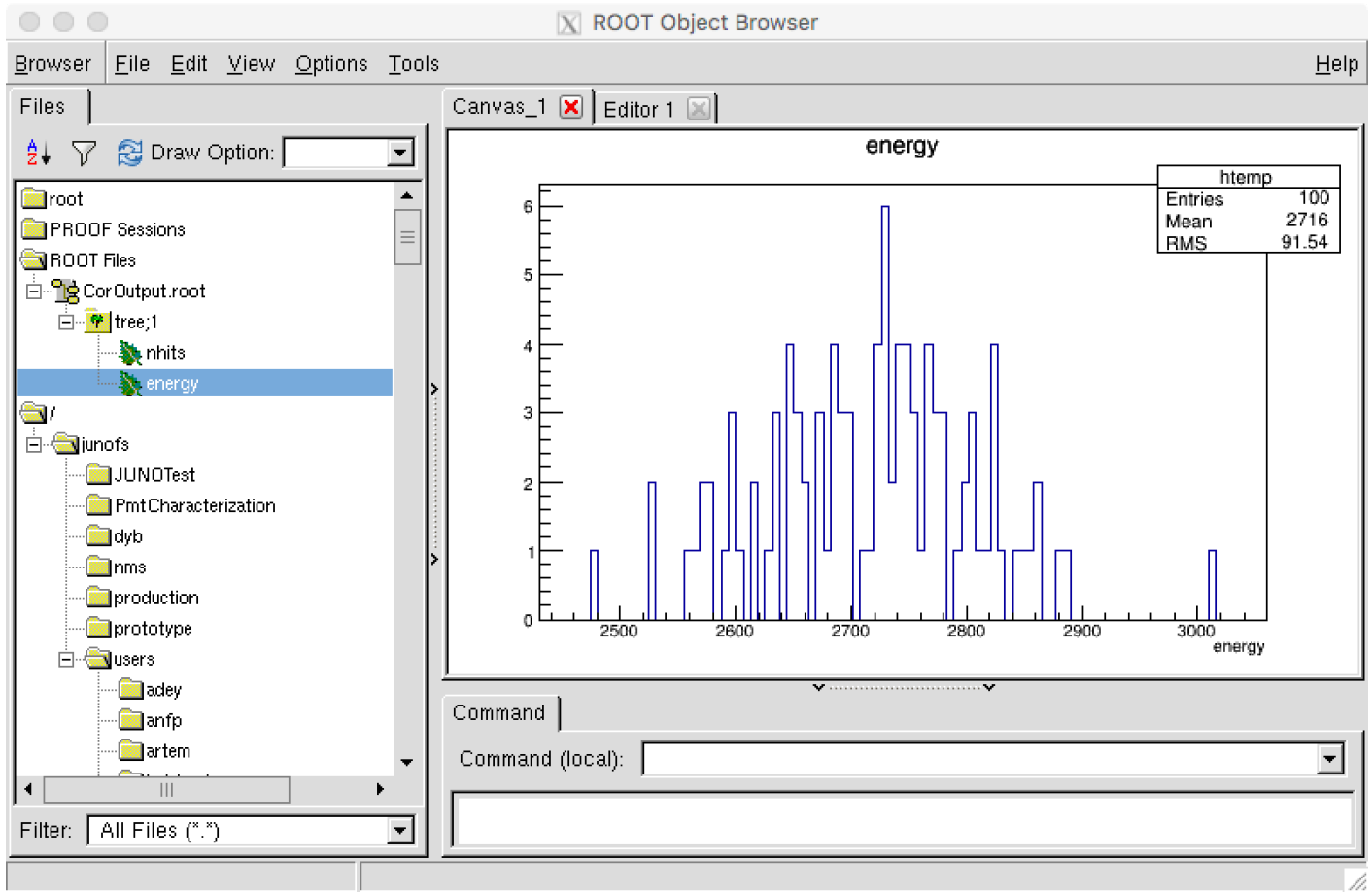
```
 1 import Sniper
 2 task = Sniper.Task("task")
 3 task.asTop()
 4 task.setLogLevel(3)
 5
 6 import EDMTutorial
 7 alg = task.createAlg("CorAnaAlg/CorAnaAlg")
 8
 9 import BufferMemMgr
10 bufMgr = task.createSvc("BufferMemMgr")
11
12 import RootWriter
13 task.property("svcs").append("RootWriter")
14 rw = task.find("RootWriter")
15 rw.property("Output").set({"FILE1": "CorOutput.root"})
16
17 import RootIOSvc
18 riSvc = task.createSvc("RootInputSvc/InputSvc")
19 task.setLogLevel(0)
20 inputFileList = ["sample_detsim.root", "sample_calib.root", "sample_rec.root"]
21 riSvc.property("InputFile").set(inputFileList)
22
23 task.setEvtMax(-1)
24 task.show()
25 task.run()
~
```

Run the Job

cd share

$ python runCA.py

$ root.exe CorOutput.root



Summary

- Brief introduction to Event Data Model
- Go through 5 Examples
  - **Define New Event Data Model Classes with XOD**
  - **Retrieve Event Data from the Buffer with the path**
  - **Create Event Data and put it into the Buffer with the path**
  - **Output flat TTree/TNtuple**
  - **Event Data Navigation**