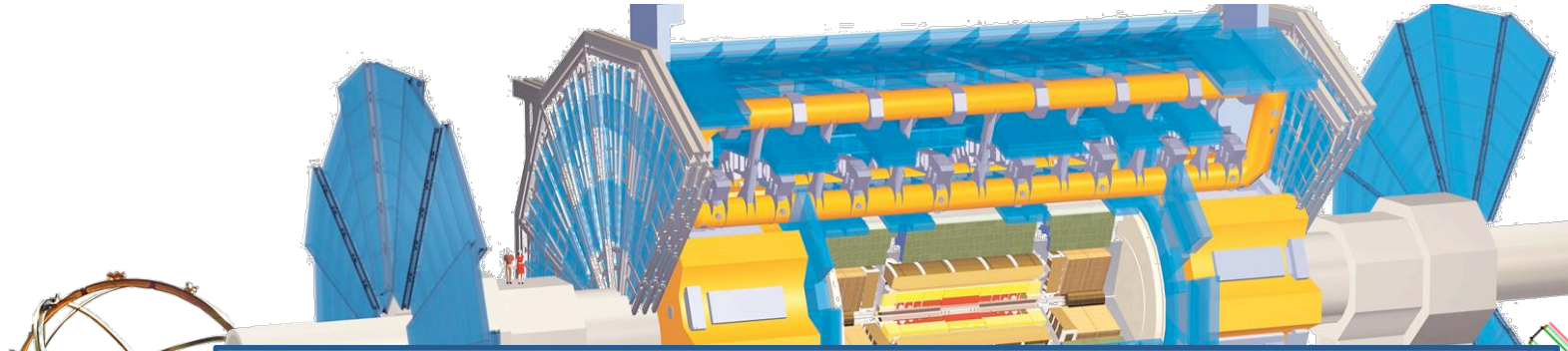
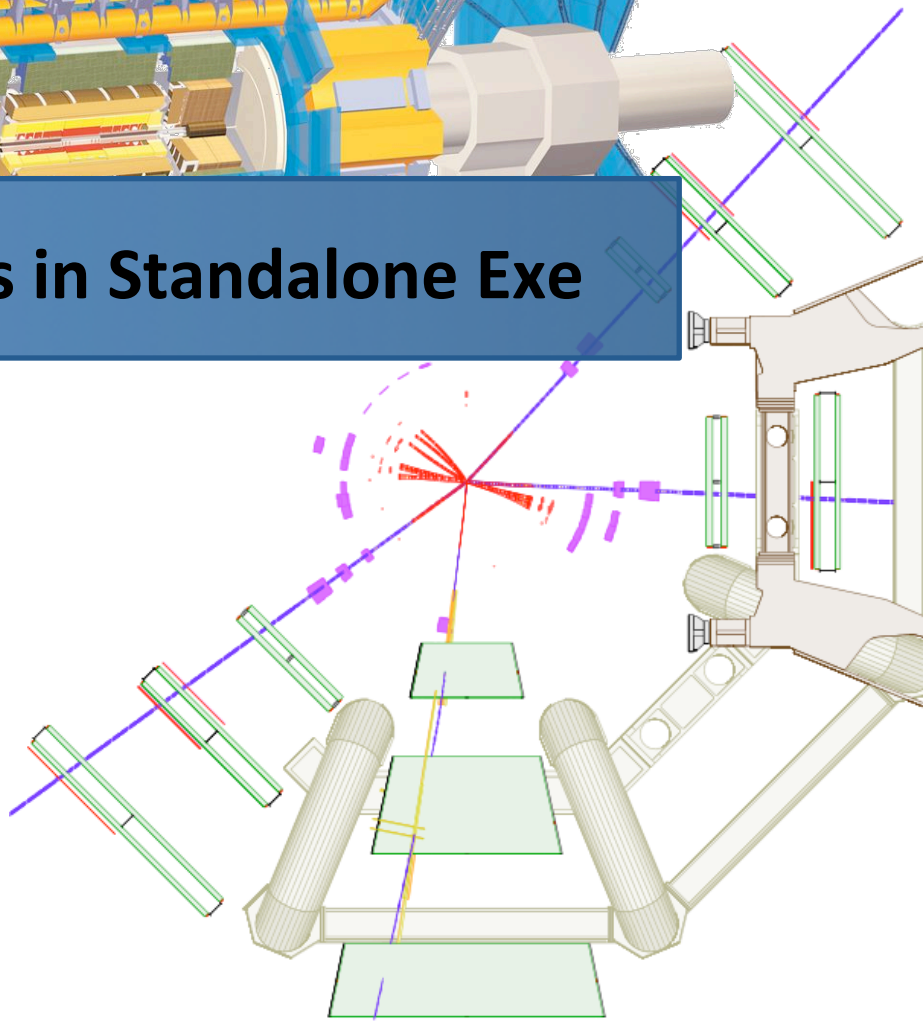


Using Components in Standalone Exe

Will Buttinger



Purpose of this talk

- Get you to think about the standalone and interactive analysis use cases
  - It is a **hard requirement** that users can use components independently of the standard framework (batch processing) setup
- Much is already possible, but many small improvements would go a long way
- Hopefully you will appreciate this perspective, and keep it in mind when you develop core framework code

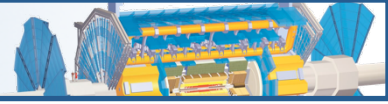
The analysis user perspective

- We can assume everybody knows ROOT
  - I will focus on c++ but everything needs to be possible similarly in python
- Reading a file in ROOT:

```
TFile f("my.file.root");
TTree* t = static_cast<TTree*>(f.Get("MyTree"));
Int myBranch(0);
T->SetBranchAddress( "myBranch", &myBranch );
For(int i=0;i<t->GetEntries();i++) {
   t->GetEntry(i);
   cout << myBranch << endl;
}
```

- Works on the ROOT prompt – interactive analysis possible
- If I destruct the TFile and the TTree, I have cleaned up everything
- No standard output messages from TFile or TTree

Standalone analysis of xAOD data format



- Attila wrote an xAOD::TEvent class that knows how to read xAOD from TFile
  - Many standalone applications use this
  - No unnecessary messages from TEvent: feels *light*

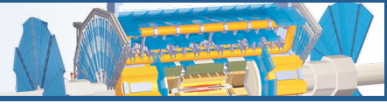
```
TFile f("my.xaod.root");
xAOD::TEvent evt;
evt.readFrom(&f);
For(int i=0;i<evt.getEntries();i++) {
   evt.getEntry(i);
   const xAOD::EventInfo_v1* ei = 0;
   evt.retrieve(ei, "EventInfo");
   cout << ei->eventNumber() << endl;
}
```

Can we do similar with athena components?

Minimum components required

- An EventSelector
  - Give it the list of input files
- A Storegate
  - Will retrieve input data from this
- The ApplicationMgr
  - If not initialized, then all services retrieved from ServiceHandles are not automatically initialized, and components are set up to handle that scenario!
  - If we want to be able to live without the AppMgr, we would have to do some work on our code

Bootstrapping Gaudi



- Found Gaudi::createApplicationMgr(), but couldn't use it at the ROOT prompt ☹
- Put it into a static method in a class, and created a dictionary for that class
  - Can now call AAH::initGaudi() straight from the prompt
- Also found this *had* to be done before any component is created
  - If create another component first, seem to get a bogus MessageSvc, which prevents the ApplicationMgr from initializing

(some of the) code from AAH::initGaudi method:

```
IAppMgrUI* theApp = Gaudi::createApplicationMgr();
…
SmartIF<IProperty> propMgr(theApp);
propMgr->setProperty( "JobOptionsType", "NONE" );
propMgr->setProperty("EventLoop","MinimalEventLoopMgr");
…
theApp->configure();
theApp->initialize()
```

The minimal gaudi environment

- What that gives us:

Silencing these messages is hard
(cannot make them all go away!)

```
root [0] AAH::initGaudi()
ApplicationMgr    SUCCESS
=======================================================================================
                                                    Welcome to ApplicationMgr (GaudiC
                                              running on pckk on Thu Sep 22 17:07:53 201
=======================================================================================
ApplicationMgr       INFO Application Manager Configured successfully
ApplicationMgr       INFO Application Manager Initialized successfully
(class IAppMgrUI *) 0x9eda110
root [1] for(auto s : Gaudi::svcLocator()->getServices()) cout << s->name() << endl
MessageSvc
JobOptionsSvc
AppMgrRunable
MinimalEventLoopMgr
IncidentSvc
```

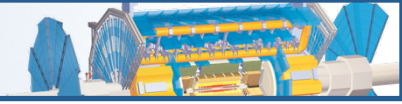
Do we really need these two?!

Minimal code

- This will run as a ROOT macro!

```
AAH::initGaudi();

ServiceHandle<IEvtSelector> evtSel("Athena::xAODEventSelector/MySelector","");
ServiceHandle<StoreGateSvc> evtStore("StoreGateSvc/MyStore_StoreGateSvc","");
AAH::setProperty( evtSel, "InputCollections", ”['my.xaod.root']”);
AAH::setProperty( evtSel, “EvtStore”, evtStore.typeAndName() );

IEvtSelector::Context* ctx = 0;
evtSel->createContext(ctx);

int numEvents = dynamic_cast<ICollectionSize*>(&*evtSel)->size();
for(int i=0;i<10;i++) {
  dynamic_cast<IEventSeek*>(&*evtSel)->seek(i);
  evtSel->next(*ctx);
  evtStore->loadEventProxies();
  const xAOD::EventInfo_v1* evtInfo = 0;
  evtStore->retrieve( evtInfo, "EventInfo");
  std::cout << "EventNumber = " << evtInfo->eventNumber() << std::endl;
  evtStore->clearStore();
}
```

Demo



- It runs, but with annoying amount of messaging
  - I was able to suppress almost all of this (not shown here) but it was not easy!!
    - E.g. some services print message before AthService::initialize call, so can't silence that with OutputLevel property

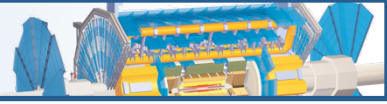
```
bash-3.2$ root demo2.C
root [0]
Processing demo2.C...
ApplicationMgr    SUCCESS
================================================================================================================================================
                                                Welcome to ApplicationMgr (GaudiCoreSvc v3r4)
                                          running on pckk on Thu Sep 22 16:00:15 2016
================================================================================================================================================
ApplicationMgr       INFO Application Manager Configured successfully
ApplicationMgr       INFO Application Manager Initialized successfully
ClassIDSvc           INFO Initializing ClassIDSvc - package version CLIDComps-00-06-18-03
ClassIDSvc           INFO  getRegistryEntries: read 1610 CLIDRegistry entries for module ALL
AthDictLoaderSvc     INFO in initialize...
AthDictLoaderSvc     INFO acquired Dso-registry
MySelector           INFO Selector configured to read [1] file(s)...
xAOD::Init                INFO    Environment initialised for data access
MySelector           INFO Using DEFAULT xAOD access mode (usually same as CLASS mode)
ProxyProviderSvc     INFO Initializing ProxyProviderSvc - package version SGComps-00-02-01
InputMetaDataStore   INFO Initializing InputMetaDataStore - package version StoreGate-03-05-01-02
MyStore_StoreGa...   INFO Initializing MyStore_StoreGateSvc - package version StoreGate-03-05-01-02
ActiveStoreSvc       INFO Initializing ActiveStoreSvc - package version StoreGate-03-05-01-02
StoreGateSvc         INFO Initializing StoreGateSvc - package version StoreGate-03-05-01-02
ClassIDSvc           INFO  getRegistryEntries: read 4037 CLIDRegistry entries for module ALL
MySelector          ERROR No CLID for class DataVector<xAOD::L2StandAloneMuon_v1> , cannot read HLT_xAOD__L2StandAloneMuonContainer_M
MySelector          ERROR No CLID for class DataVector<xAOD::TrigRNNOutput_v1> , cannot read HLT_xAOD__TrigRNNOutputContainer_TrigTRT
MySelector        WARNING The following AuxStore types are not directly accessible (missing CLID, possibly from schema evolution): xA
uonAuxContainer_v1, xAOD::MissingETAuxAssociationMap_v1, xAOD::MuonAuxContainer_v2, xAOD::TrackParticleAuxContainer_v2, xAOD::TrigRNN
r_v1,
EventPersistenc...   INFO Added successfully Conversion service:Athena::xAODCnvSvc
EventNumber = 187901
EventNumber = 187906
EventNumber = 187801
EventNumber = 187953
EventNumber = 187960
EventNumber = 185533
EventNumber = 187864
EventNumber = 180559
EventNumber = 185523
EventNumber = 185506
root [1]
```

Minimal code

- Name of StoreGateSvc had to contain ‘StoreGateSvc’ or ‘EventStore’ in it to get the correct StoreID type (otherwise selector ignores it)

```
AAH::initGaudi();

ServiceHandle<IEvtSelector> evtSel("Athena::xAODEventSelector/MySelector","");
ServiceHandle<StoreGateSvc> evtStore("StoreGateSvc/MyStore_StoreGateSvc","");
AAH::setProperty( evtSel, "InputCollections", ”['my.xaod.root']”);
AAH::setProperty( evtSel, “EvtStore”, evtStore.typeAndName() );

IEvtSelector::Context* ctx = 0;
evtSel->createContext(ctx);

int numEvents = dynamic_cast<ICollectionSize*>(&*evtSel)->size();
for(int i=0;i<10;i++) {
  dynamic_cast<IEventSeek*>(&*evtSel)->seek(i);
  evtSel->next(*ctx);
  evtStore->loadEventProxies();
  const xAOD::EventInfo_v1* evtInfo = 0;
  evtStore->retrieve( evtInfo, "EventInfo");
  std::cout << "EventNumber = " << evtInfo->eventNumber() << std::endl;
  evtStore->clearStore();
}
```

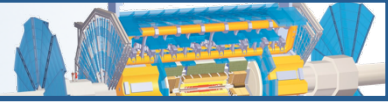
Minimal code



- Setting properties on service before it is initialized was not ‘straightforward’
  - Need to add properties to the JobOptionsSvc: wrote a helper method for this

```
AAH::initGaudi();

ServiceHandle<IEvtSelector> evtSel("Athena::xAODEventSelector/MySelector","");
ServiceHandle<StoreGateSvc> evtStore("StoreGateSvc/MyStore_StoreGateSvc","");
AAH::setProperty( evtSel, "InputCollections", "['my.xaod.root']");
AAH::setProperty( evtSel, "EvtStore", evtStore.typeAndName() );

IEvtSelector::Context* ctx = 0;
evtSel->createContext(ctx);

int numEvents = dynamic_cast<ICollectionSize*>(&*evtSel)->size();
for(int i=0;i<10;i++) {
  dynamic_cast<IEventSeek*>(&*evtSel)->seek(i);
  evtSel->next(*ctx);
  evtStore->loadEventProxies();
  const xAOD::EventInfo_v1* evtInfo = 0;
  evtStore->retrieve( evtInfo, "EventInfo");
  std::cout << "EventNumber = " << evtInfo->eventNumber() << std::endl;
  evtStore->clearStore();
}
```

Minimal code



- Using ServiceHandles – awkward if I need to utilize multiple interfaces
  - Also why is the second argument in constructor necessary (name of caller)?

```
AAH::initGaudi();

ServiceHandle<IEvtSelector> evtSel("Athena::xAODEventSelector/MySelector","");
ServiceHandle<StoreGateSvc> evtStore("StoreGateSvc/MyStore_StoreGateSvc","");
AAH::setProperty( evtSel, "InputCollections", "['my.xaod.root']");
AAH::setProperty( evtSel, "EvtStore", evtStore.typeAndName() );

IEvtSelector::Context* ctx = 0;
evtSel->createContext(ctx);

int numEvents = dynamic_cast<ICollectionSize*>(&*evtSel)->size();
for(int i=0;i<10;i++) {
  dynamic_cast<IEventSeek*>(&*evtSel)->seek(i);
  evtSel->next(*ctx);
  evtStore->loadEventProxies();
  const xAOD::EventInfo_v1* evtInfo = 0;
  evtStore->retrieve( evtInfo, "EventInfo");
  std::cout << "EventNumber = " << evtInfo->eventNumber() << std::endl;
  evtStore->clearStore();
}
```

What services exist?

- Can I easily see the type of the service?

```
root [1] for(auto s : Gaudi::svcLocator()->getServices()) cout << s->name() << endl
MessageSvc
JobOptionsSvc
AppMgrRunable
MinimalEventLoopMgr
IncidentSvc
ClassIDSvc
AthDictLoaderSvc
IoComponentMgr
EventPersistencySvc
ProxyProviderSvc
MySelector
InputMetaDataStore
MyStore_StoreGateSvc
StoreGateSvc
ActiveStoreSvc
Athena::xAODCnvSvc
```

Ideally should get a unique ProxyProviderSvc

Other stores created too!! Not got unique names

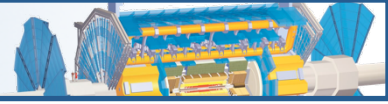
Something is expecting the store to be called StoreGateSvc!!

Minimal code

- Use an AthenaEventLoopMgr to manage some tasks
  - Is this best way? Still adds another component user must think about

```
AAH::initGaudi();

ServiceHandle<IEvtSelector> evtSel("Athena::xAODEventSelector/MySelector","");
ServiceHandle<StoreGateSvc> evtStore("StoreGateSvc/MyStore_StoreGateSvc","");
AAH::setProperty( evtSel, "InputCollections", "['my.xaod.root']");
AAH::setProperty( evtSel, "EvtStore", evtStore.typeAndName() );

IEvtSelector::Context* ctx = 0;
evtSel->createContext(ctx);

int numEvents = dynamic_cast<ICollectionSize*>(&*evtSel)->size();
for(int i=0;i<10;i++) {
  dynamic_cast<IEventSeek*>(&*evtSel)->seek(i);
  evtSel->next(*ctx);
  evtStore->loadEventProxies();
  const xAOD::EventInfo_v1* evtInfo = 0;
  evtStore->retrieve( evtInfo, "EventInfo");
  std::cout << "EventNumber = " << evtInfo->eventNumber() << std::endl;
  evtStore->clearStore();
}
```

Minimal code



- Use an AthenaEventLoopMgr to manage some tasks
  - Interface doesn't seem that nice (seek + nextEvent??)

```
AAH::initGaudi();

ServiceHandle<IEvtSelector> evtSel("Athena::xAODEventSelector/MySelector","");
ServiceHandle<StoreGateSvc> evtStore("StoreGateSvc/MyStore_StoreGateSvc","");
ServiceHandle<IEventProcessor> evtLoop("AthenaEventLoopMgr/MyLoop","");
AAH::setProperty( evtSel, "InputCollections", "['my.xaod.root']");
AAH::setProperty( evtSel, "EvtStore", evtStore.typeAndName() );

//Connect the selector and storegate to the processor
AAH::setProperty( evtLoop, "EvtSel", evtSel.typeAndName());
AAH::setProperty( evtLoop, "EvtStore", evtStore.typeAndName());
//And then ensure configured for interactive work
AAH::setProperty( evtLoop,"ClearStorePolicy","BeginEvent");

int numEvents = dynamic_cast<ICollectionSize*>(&*evtSel)->size();
for(int i=0;i<10;i++) {
  dynamic_cast<IEventSeek*>(&*evtLoop)->seek(i);evtLoop->nextEvent(i+1);
  const xAOD::EventInfo_v1* evtInfo = 0;
  evtStore->retrieve( evtInfo, "EventInfo");
  std::cout << "EventNumber = " << evtInfo->eventNumber() << std::endl;
}
```

Further questions/comments

- How do I clean all this up?
  - Deleting services? Deleting tools?
- Reading two files concurrently ~ doable, but worry about services cross-talking

- If you make a typo in ROOT cling prompt, ROOT falls over badly and then can't even recover itself
  - I have no idea if we can make Gaudi more robust to this so that ROOT is less unforgiving

- Using handles, don't get nice dictionary help from ROOT
  - Even auto-complete ends up being anti-helpful: doesn't think you want to dereference a handle

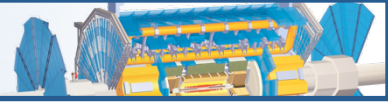
- Similar challenges faced with tools as with services
  - Creating them
  - Configuring them
  - Destroying them
  - Connecting them

Reading POOL files

- Extra pieces needed for POOL reading
  - Must use EventLoopMgr. Extra properties (couldn't the selector set these up?)

```
AAH::initGaudi();

AAH::addPropertyToCatalogue("EventPersistencySvc","CnvServices","['AthenaPoolCnvSvc']");
AAH::addPropertyToCatalogue("ProxyProviderSvc","ProviderNames","['AthenaPoolAddressProviderSvc']");

ServiceHandle<IEvtSelector> evtSel("EventSelectorAthenaPool/MySelector","");
ServiceHandle<StoreGateSvc> evtStore("StoreGateSvc/MyStore_StoreGateSvc","");
ServiceHandle<IEventProcessor> evtLoop("AthenaEventLoopMgr/MyLoop","");
AAH::setProperty( evtSel, "InputCollections", "['my.xaod.root']");
//AAH::setProperty( evtSel, "EvtStore", evtStore.typeAndName() );

//Connect the selector and storegate to the processor
AAH::setProperty( evtLoop, "EvtSel", evtSel.typeAndName());
AAH::setProperty( evtLoop, "EvtStore", evtStore.typeAndName());
//And then ensure configured for interactive work
AAH::setProperty( evtLoop,"ClearStorePolicy","BeginEvent");

int numEvents = dynamic_cast<ICollectionSize*>(&*evtSel)->size();
for(int i=0;i<10;i++) {
  dynamic_cast<IEventSeek*>(&*evtLoop)->seek(i);evtLoop->nextEvent(i+1);
  const xAOD::EventInfo_v1* evtInfo = 0;
  evtStore->retrieve( evtInfo, "EventInfo");
  std::cout << "EventNumber = " << evtInfo->eventNumber() << std::endl;
}
```

Running an algorithm



- No dictionary, so use factory (also direct constructor is ‘unpleasant’: svcLocator)

```
AAH::initGaudi();

AAH::addPropertyToCatalogue("EventPersistencySvc","CnvServices","['AthenaPoolCnvSvc']");
AAH::addPropertyToCatalogue("ProxyProviderSvc","ProviderNames","['AthenaPoolAddressProviderSvc']");

ServiceHandle<IEvtSelector> evtSel(”EventSelectorAthenaPool/MySelector","");
ServiceHandle<StoreGateSvc> evtStore("StoreGateSvc/MyStore_StoreGateSvc","");
ServiceHandle<IEventProcessor> evtLoop("AthenaEventLoopMgr/MyLoop","");
AAH::setProperty( evtSel, "InputCollections", ”['my.evnt.root']”);

AAH::setProperty( evtLoop, "EvtSel", evtSel.typeAndName());
AAH::setProperty( evtLoop, "EvtStore", evtStore.typeAndName());
AAH::setProperty( evtLoop,"ClearStorePolicy","BeginEvent");

IAlgorithm* myAlg = AAH::createAlgorithm(“xAODMaker::xAODTruthCnvAlg/myAlg”);
AAH::setProperty( myAlg , “AODContainerName”, “GEN_EVENT”);
AAH::setProperty( myAlg , “EvtStore”, evtStore.typeAndName() );
myAlg->initialize();

int numEvents = dynamic_cast<ICollectionSize*>(&*evtSel)->size();
for(int i=0;i<10;i++) {
  dynamic_cast<IEventSeek*>(&*evtLoop)->seek(i);evtLoop->nextEvent(i+1);
  myAlg->execute();
  std::cout << evtStore->dump() << std::endl;
}
```

Notes

- AAH code lives here:
  - https://svnweb.cern.ch/trac/atlasoff/browser/Control/AthAnalysisBaseComps/trunk/
- Wrapped the EvtLoop, EvtStore, and EvtSelector combo into a helper class called POOL::TEvent
  - https://svnweb.cern.ch/trac/atlasoff/browser/PhysicsAnalysis/POOLRootAccess/trunk/
  - Available in AthAnalysisBase and release 21
  - Includes messaging suppression tricks I had to play
  - Includes python bindings for use in python (see ATN test in package)