Report of the
AOD Format Task Force

Ketevi Assamagan, Kevin Black, Paolo Calafiura,
Davide Costanzo, Kyle Cranmer, Andrea Dell' Acqua,
Amir Farbin, Peter van Gemmeren, Stephen Gowdy,
Roger Jones, Tommaso Lari, David Malon, Frank Paige,
Giacomo Polesello, Scott Snyder
Ex-officio: Dario Barberis, Fabiola Gianotti, Ian Hinchliffe,
David Quarrie

Editor: Paolo Calafiura
Last modified: 06. Dec. 2006
First created: 10. Nov. 2006

Editor: Paolo Calafiura
Last modified: 18. Dec. 2006
First created: 10. Nov. 2006

AOD Format Task Force
Draft Report

1. Introduction

The Analysis Object Data (AOD) are produced by ATLAS reconstruction and are the main input for most analyses. AOD, like the Event Summary Data (ESD, the other main output of reconstruction) are written as POOL files and are readable from Athena, and, to a limited extent, from ROOT. The AOD typical size, processing speed, and their relatively complex class structure and package dependencies, make them inconvenient to use for most interactive analysis.

According to the computing model, interactive analysis will be based on Derived Physics Data (DPD), a user-defined format commonly produced from the AOD. As of release 12.0.3 it is common practice to write DPD as Athena-aware Ntuples (AANT) in ROOT. In an effort to organize and standardize AANT, we introduced the Structured Athena-aware Ntuple (SAN), an AANT containing objects that behave, as much as it is allowed by ROOT interpreter limitations, as their AOD counterparts[SAN].

Recently it was proposed to extend SAN functionality beyond DPD implementation [SAN-AOD]. SAN objects would be used as AOD objects. The TOB formed our task force with the mandate to
perform a technical evaluation of the two proposals, one based upon the existing AOD classes and architecture, the other upon Structured Athena-Aware Ntuples.
[...]
Criteria for the evaluation should include I/O performance, support for schema evolution, suitability for end user analysis and simplicity.

Editor: Paolo Calafiura
Last modified: 18. Dec. 2006
First created: 10. Nov. 2006

2. Baseline Analysis Model

Analysis Streams

As described in the Computing TDR [CM], the standard reconstruction job will produce three data streams that will be used for analysis in Athena:

- ESD (Event Summary Data): written using POOL, this data stream contains detector-level calibrated quantities such as Calorimeter cells and ID hits, plus the reconstruction objects built from them (e.g. clusters, tracks). ESD target size is ~0.5 MB/event, 12.0.x size ~1.5MB/event. Hosted by Tier-1s, ESD will be available for high-statistics processing only via managed productions about ten times/year[1].
- AOD (Analysis Object Data): also written using POOL, this data stream contains high-level physics object such as particles, as well as clusters and tracks associated with particles. Target size ~100KB/event, 12.0.x size ~200KB/event (40% of which is Truth). Hosted by Tier-2s, AODs will be available for user-initiated processing with a turnaround time of days.
- TAG: a tag database that allows to quickly select events in ESD and AOD based on a summary of the information in the AOD.

Besides these three centrally produced streams, physics groups and individuals are expected to define several streams of

- DPD (Derived Physics Data): AOD subsample reduced in size for example applying stricter event selection (*skimming*), reducing in size the information per object (*slimming*) and dropping unwanted (elements of) data objects (*thinning*). DPD may also contain additional user-defined data, such as composite particle object. DPD samples will be used for interactive analysis in ROOT and they are expected to sit comfortably in the disk of a laptop. Typical size will be 10KB/event with typical processing rate of ~1 kHz. The baseline technology to write DPD data is the Athena-Aware Ntuple (AANT), a ROOT tuple containing tag information that can be used for fast event selection. DPDs are produced in Athena jobs, using a common framework like EventView [EV]. Physics coordination is currently evaluating the definition of a few commonly defined DPD streams (e.g. for High-pt physics [HighPT EV]).

---

1 Calibration will be performed using ESD accessible, just for this purpose, from CERN CAF.

EDM Performance Optimizations

Past experience shows that AOD access speed is the main factor that will determine AOD acceptance by the user community. AOD size is also constrained by the Computing Model. We expect to be able to optimize AOD access speed and to reduce AOD size back to its Computing Model target using Transient/Persistent (T/P) separation, an approach to object persistence that allows optimization (and modification) of the object's representation used in the persistent store, without affecting their public, “transient” representation, used everywhere else.

Analysis Work-flow

After the initial detector and reconstruction debugging work in 2007/2008, physicists performing their analysis will probably begin their work looking at one of the DPD defined by their group. To improve their analysis performance they will go back to AODs to tune particle reconstruction and DPD definition. They will also go back to ESDs to skim interesting events and reprocess them, or for studies of systematics (the latter hopefully may be performed on a subsample of ESDs).

Thanks to the ongoing merging between ESD and AOD object interfaces, analysis tools for tasks like electron/photon identification, b-tagging, and jet corrections should run on either ESD or AOD. From the analysis point of view, ESD and AOD will differ mainly in the level of detail provided by a largely overlapping set of data objects. Once we acquire enough experience with the analysis it is possible that ESD will be confined to specialized tasks such as calibration.

Convergence between DPD and AOD is not so easy, because they are based on somewhat different technologies (ROOT and POOL), and because in the current model we expect most users to look at DPDs interactively from ROOT, and to process AOD in batch mode using Athena. AANTs are currently structured as “flat” Ntuples containing Plain Old Data types (basically ints, floats and arrays thereof) and the tools to analyze an Electron in AOD or the electron variables in a AANT are completely different.

Editor: Paolo Calafiura
Last modified: 18. Dec. 2006
First created: 10. Nov. 2006

3. SAN and its use as DPD and AOD

Structured Athena-aware Ntuples [SAN] were introduced to bridge the gap between AANT and AOD based analysis [EV]. If one could store AOD Electron objects in a branch of the AANT, the AANT analysis code would become easier to develop, maintain and hopefully it could be shared with AOD analysis tools. Unfortunately, due to limitations of ROOT interpreter and dictionary, or, if you will, to the complexity of the particle class structure which is used by most AOD classes, it is currently impossible to use an AOD Electron stored in a AANT branch. For this reason a SAN Electron, while sharing much of the same IParticle-like interface of an AOD Electron, does not implement all its aspects; and in particular the Navigable functionality[Navigables].

SAN as DPD

SAN-based DPD is an attempt to provide the essence of AOD functionality in a lightweight package. Besides "helping" ROOT understand our data model, there are other reasons to simplify the Particle classes. In a "laptop analysis" scenario it is important to be able to look at a SAN file without the need to load scores of ATLAS libraries into ROOT. Last but not least, simplifying the object structure should bring a non-negligible speed advantage. In our evaluation described below we found that the current prototype meets these goals.

SAN as AOD

Using SAN as AOD would allow to analyze AOD data directly from ROOT. Using a single set of classes for AOD and DPD would ease the maintenance load of PAT developers (although developing a set of common classes would initially be harder). A common set of classes would also allow to share analysis code between DPD and AOD and would simplify the job of producing a DPD from the AOD. The requirements for using SAN as an AOD format are more stringent and are not entirely met by the current SAN prototype[BNL PAT Meeting]:

1. As we said, ROOT can not handle the Navigable part of our ParticleBase class. A SAN-based AOD must find a way to store this information in a way that is

transparent to ROOT, while making it fully accessible from Athena.

2. The current SAN implementation would not support transient/persistent separation and in general the Athena converter-based I/O mechanism. Architectural considerations aside, this would create a conflict between the usability of AOD objects and the performance of persistence. Without T/P separation we would also lose our current baseline solution for schema evolution and would have to rely on ROOT automatic schema evolution or on ROOT streamers[1].

In conclusion using the current SAN prototype as AOD would make ROOT-based analysis simpler but it would introduce some implementation challenges and risks when it comes to AOD access from Athena.

SAN as Persistent AOD

Persistent AOD classes could be equipped, to the extent that ROOT CINT parser would allow it, with an interface equivalent to their corresponding transient AOD classes [BNL PAT meeting]. If, in addition, one could use POOL "placement hints" to give POOL AOD and DPD files an event structure that is convenient to use from ROOT (namely with a single event TTree with one branch per data object), we could in principle achieve almost the same functionality provided by SAN as AOD, without violating the T/P separation of AODs[TPAOD]. Following the discussions in this TF, an initial prototype of this enhanced pAOD (or "pAOD as DPD") was introduced[pAODROOTAccess].

Comparing and contrasting the two prototypes a third solution emerged[2], namely to use the current SAN data classes to implement the persistent layer of the current POOL-based AOD ("SAN as pAOD"). SAN classes would keep their current AOD-like interface for ROOT analysis. We believe, although a realistic test has yet to be performed, that we can structure the POOL AODs as a single event TTree with one branch per AOD container, keyed using StoreGate identifiers. As a consequence an AOD file can be accessed directly from ROOT as SAN and it would have the same structure and most of the same content that AOD provide in Athena.

The biggest advantages of this approach would be on the Athena side. Using the standard T/P conversion mechanism, the SAN objects in AOD and DPD could be

---

1 Or rather the souped-up streamers capable to write in split-mode currently being developed by Scott

2 reportedly from the corridors of BNL physics division...

turned in their transient AOD counterparts. From Athena one could run the same analysis code on ESD, AOD, and DPD data and use common tools to produce personalized AOD and DPD streams.

Using the same classes for the persistent representation of AOD and DPD and for Ntuple-style analysis should reduce maintenance effort and promote analysis reproducibility.

The "SAN as pAOD" concept solves, almost by definition, the Athena architecture compatibility issues described in the "SAN as AOD" section.

The "SAN as pAOD" would still have the issue of carrying a non-negligible amount of data used to build transient AODs and not easily usable in ROOT.

StoreGate persistable references (Data/ElementLink) and ROOT persistable references (TRefs) do not work together out of the box. To be able to dereference persistable references in ROOT and Athena one can simply add them in pairs to the SAN object (a SAN Electron would then have both a TRef and an ElementLink to its Track). This would not only waste space, but it would put on the user the burden of keeping this references in sync. A far better solution could be to keep only one of the two in the SAN objects either using TRef as the persistent representation of an ElementLink, or by adding to the persistent ElementLink_p class the ability to navigate in ROOT to the TBranch containing the pointed-to object. We do not have at the moment a detailed design of either of this possibilities but they both seem implementable[1].

"SAN as pAOD" also have a few specific issues. One, that we mentioned already, is that we have yet to demonstrate that a POOL file can be structured in a way that allows easy and natural browsing from ROOT.

Keeping the interfaces of transient AOD and persistent AOD (SAN) objects in sync will introduce some extra maintenance work for PAT developers. In Athena, analysis code will have to be structured carefully if we want to use it for both AOD and DPD objects. In particular it should handle gracefully the missing functionality in SAN (Navigables, but also thinned DataVectors, "dropped" pointers, etc).

Analysis scripts in ROOT should be insulated from schema evolution of the pAOD classes. This should be relatively straightforward as long as the scripts rely only on the SAN objects interface.

---

1 If unsurmountable problems were to arise, adding both the persistent representation of the SG link and the Tref to the pAOD objects would be an (ugly) fall-back solution.

Potentially the biggest problem of the “SAN as pAOD proposal” could be the handling of complex T/P mappings. In particular the new Marcin/RD scheme to support specific containers like TrackParticle's [New T/P], will write the elements of, say, a DataVector<IParticle> scattered over a number of ROOT collections. The DataVector is rebuilt on the fly by the T/P layer when the object is read back from disk. Hiding this operation behind the interface of the SAN object used in ROOT will not be trivial.

4. Evaluation of DPD Prototypes

Usability tests are discussed separately for the two DPD formats investigated: the SAN and the current implementation of pAOD classes. Since the SAN prototype is more complete and was made available earlier, more extensive tests were performed on it. According to the recommendations of this document, the SAN prototype will provide the basis for the persistent representation of the AOD; thus, the two approaches will eventually be merged.

Producing DPDs from Athena

The SAN can be produced as a DPD from Athena via an extension of the Athena aware Ntuple production mechanism. This was implemented in the UserAnalysis package which calls a series of Athena tools and services to extract the relevant information from the AOD and creates the SAN. Producing the SAN from the AOD using the default mechanism is a simple Athena job with the provided jobOptions. Small modifications of the existing objects in the SAN such as addition of simple types is fairly straightforward. The creation of links between objects (e.g. electron to track) is somewhat more involved because of the need to create a persistent representation of C++ pointers. In general we feel that extending the DPD from Athena in non trivial ways would be more than what the average user would feel comfortable to do. If such modification is intended for general use, a simpler user interface should be provided.

Analyzing DPDs in ROOT (and Athena)

Analyzing SAN

Tests were performed on a file containing 4000 ttbar events. The following classes were available in the file as separate branches of a single tree:

The primary vertex: VxPrimaryCandidate

Calorimeter clusters: LArClusterEM37, LArClusterEM, LarClusterEMSofte, LArClusterEMgam35, LArClusterEMgam, CaloCalTopoCluster, EMTopoCluster, CombinedCluster

Tracks: TrackParticleCandidate, StacoTrackParticles, MuonboyMuonSpectroOnlyTrackParticles, MuonboyTrackParticles, MuTagTrackParticles, MeasuredPerigee

Particles, jets, missing Et: ElectronCollection, PhotonCollection, StacoMuonCollection, TauJetCollection, ConeTowerParticleJets, MET_Final_Muonboy

MC truth: SpclMC, nonSpclMC, MET_Truth, ConeTruthParticleJets

As mentioned in previous sections, the infrastructure to access to ROOT trees from Athena has not yet been implemented. In order to access to the file from ROOT, the following operations are necessary:

On a machine with an Athena installation: After performing the Athena setup, the following macro should be executed from the ROOT prompt:

```
Cintex::Enable();
gSystem->Load("libUserAnalysisEventDict");
gSystem->Load("libUserAnalysisEvent");
gSystem->Load("libMathCore");
```

```
gSystem->SetIncludePath("-I
afs/cern.ch/user/t/tlari/scratch0/12.3.0/PhysicsAnalysis/AnalysisCommon/UserAnalysisEvent/User
AnalysisEvent-00-00-39  -I/afs/cern.ch/sw/lcg/external/clhep/1.9.2.2/slc3_ia32_gcc323/include");
```

On a machine without Athena: The feasibility of accessing to the SAN file without Athena installed must still be fully investigated. The feasibility of compiling the SAN specific code (a couple of packages), which should work also with other operating systems and compilers, has not been tested yet.

The browsability of the information contained in the SAN is quite good. Each class is visible from the ROOT browser as a branch having the same name as the class; the data members of each class are also displayed. One can obtain a plot of each of them by clicking on it. From the ROOT command line, one can use the methods of the EDM classes. Navigation from an electron to the associated cluster and track (stored in separated branches) is also straightforward:

```
tree>Draw("ElectronCollection.cluster().e()/(ElectronCollection.track().p())",
        "ElectronCollection.isEM() == 0","", 4000, 0);
```

The development of a more complex analysis using ROOT macros also proceeded smoothly. An example can be found in /afs/cern.ch/user/t/tlari/public/aodtf (the relevant files are Analysis.C and Analysis.h). This analysis was developed from scratch in a few days. The analysis does the following operations:

1. Reads a significant subset of the information contained in the tree. This is done by using the standard ROOT methods to access the variables of an Ntuple, the only difference with a flat Ntuple being that the variables are not *double* or *int* types but things like *std::vector<Electron>*. The list of objects which are read by the macros are the electrons, muons, photons, jets, EM and hadronic clusters, ID tracks, missing Et, and truth particles.

2. A number of histograms are filled and written to a file. Each histogram verifies the accessibility of a specific piece of information. As an example, for the electron, the histogram list includes basic kinematic information, the isEM flag, isolation, the number of B-layer, pixel, SCT, TRT, and TRT high-threshold hits, the energy of the associated cluster, and the momentum of the associated track. The access to the

properties of the other objects has been tested to a similar level of detail. The methods of the analysis EDM classes have been used throughout the code.

3. The SAN files which was used was produced without any overlap removal. This was partially implemented in the ROOT macro. Good electrons were identified and used to fill a user-defined electron container. A user-defined jet container was created by removing from the jet list those which match an identified electron. The deltaR method of the LorentzVector class was used to do the matching.

4. The truth particle container was accessed, the navigability down and up the decay chain was verified, and an angular match between reconstructed objects and MC truth was used to measure fake rates and efficiencies.

5. The tau container was accessed and various variables were retrieved. The links to the track particles were retrieved and the invariant mass of the tracks was reconstructed. The link from tau particle to tracks was very clear and concise and worked as advertised.

6. It was generally agreed that the target "analysis rate" (number of events analyzed per second) must be comparable to analysis on a flat ROOT Ntuple, and hence on the order of kHz. Informal timing tests were done on the SAN by writing simple ROOT macros to do semi-realistic analysis. Without pretense of being rigorous, our tests indicate that for "TTree::Draw-style" analysis (reading one or two particle containers at a time), SAN will perform at rates >1 kHz. When whole events are read the observed analysis rate is ~200 Hz. Our results indicate that our relatively small (300 MB) SAN file was preloaded into memory and hence do not include any disk I/O. Such caching improves analysis speed of small files regardless of the format used(flat N-tuples, SAN, AOD). For uncached files real-time execution will be significantly slower[1].

1initial tests have shown that flushing the disk cache before reading slows down I/O by a factor 3\times-20\times depending on how much of the event is read.

Object	***Average Analysis Time Per Event (ms)***	***"Analysis Rate" (kHz)***
Electron	0.63	1.6
Photon	0.21	5
Muon	0.07	14
TauJet	0.07	14
CaloTopoCluster	0.31	3
Particle Jet	0.65	1.5
RecVertex	0.05	22.2
TrackParticle	0.93	1.08
TruthParticle	0.54	1.8

Exploratory analysis

The development of this code did not present particular headaches for the user, compared to, say, a ROOT based analysis developed to access a CBNT flat Ntuple. No test of portability of the code back into Athena for an AOD-based analysis was made, but it is expected to be relatively straightforward, since the methods of AOD classes are used.

Our conclusions are that the ROOT-based analysis on SAN data is quite user-friendly. The present prototype already has most of the functionality desired for physics analysis. It also offers substantial advantages over the presently used DPD formats (mostly user-defined flat Ntuples) because of its use of AOD classes and methods. This greatly increases the possibility of using common tools and sharing pieces of code between different analysis, as well as between DPD-based and ESD/AOD-based analyses. The code is available in release 12.0.4, and users should be encouraged to try the present SAN prototype for their analysis.

pAOD based analysis

Editor: Paolo Calafiura
Last modified: 18. Dec. 2006
First created: 10. Nov. 2006

The pAOD files could be read from ROOT after the Athena setup, provided that one loads a few libraries first. While the data in the SAN were organized as separate branches of the same tree, the present pAOD implementation stores the data of each persistent class as a separate TTree. These could be displayed using the ROOT browser. The names of the trees are relatively long:

POOLContainer_Rec::TrackParticleContainer_TrackParticleCandidate

but the Athena EDM classes can still easily be recognized. Obviously this aspect could be greatly improved if the pAOD structure could be made similar to the SAN one with a single event TTree with one TBranch per data object. We understand that this will be possible using POOL placement hints available in the 12.x development releases. The data members of each object can be accessed from the browser. It is possible to use the methods of the AOD classes to plot a distribution from the ROOT command line prompt. The choice is however presently restricted only to the subset of AOD methods which are implemented.

Since objects of different types are stored in separate trees, in order to plot correlations between variables of two such objects, it is necessary to declare the two trees as friends. It is not easy to navigate from an electron to the associated track from the command line prompt.

Several of the tests that were performed for the SAN were also performed on the pAOD. Due to the partial implementation, not all of the tests were possible. In general it was felt that fundamentally the two formats could be reasonably user-friendly. However, at this time it was also clear that the pAOD interface would need to be improved. For example, although it was not difficult to follow the provided examples, the use of very complicated and long names, the necessity of declaring different trees as friends, and explicit setting up of TRefs inside the analysis code[1] are all aspects that should be improved or hidden from the end user.

1 We understand that a subsequent version of the prototype that we could not test in time address this specific problem.

5. Discussion

The Task Force endorses the goals of the original "SAN as AOD proposal", namely:

1. A common format to write AOD and DPD[1]
2. A common interface to access high-level physics objects in both AOD and DPD.
3. Easy migration of physics analysis code between Athena and ROOT[2].
4. Convenient access to AODs and DPDs from ROOT (and, of course, from Athena).
5. Lightweight implementation of high-level objects with minimal dependencies on non-analysis packages.

The technical challenge is how to achieve these goals while respecting the Athena I/O architecture (T/P separation and technology-independence of data objects).

The further constraint is that we need to recommend a solution which is implementable on a timescale of weeks, and that minimize technical risks.

Our recommendation is to deploy SAN in two stages, which should mitigate risk and provide physicists with an immediately usable solution in the first stage. Since our evaluation has demonstrated that the current SAN prototype is a very strong candidate to replace the flat AANT as the experiment's DPD format, we should provide in Athena the code to write SAN-based DPDs as soon as possible. Then we should demonstrate that a SAN-based DPD is usable from ROOT in "stand-alone mode" (without having to set up the full Athena run-time environment). At the same time we recommend that existing PAT tools such as the EventView framework are made SAN-compatible. In particular it should be possible to write SAN-based DPDs from the EventView framework. With the SAN-based DPD in production we will be able to obtain real-life experience and feedback on missing features from DPD users.

The second stage of the deployment should focus on addressing the technical issues and better understanding the risks of the "SAN as pAOD" approach. With a focused effort we believe that the main issues (persistable references, impact of

1 The difference between the two streams would then be the reduction in size achieved via event selection, DataVector thinning, etc.

2 Save for the aforementioned limitations coming from the current ROOT CINT implementation.

schema evolution on DPD analysis scripts, complex T/P mappings) can be addressed in a month and that a complete "SAN as pAOD" prototype could be delivered in the second part of January. This Task Force could then reconvene and quickly perform an evaluation of the prototype similar to the one described above. The prototype, if approved, could be quickly put in production and become part of release 13. If this, admittedly aggressive, schedule would slip, we would still have a solid SAN-based DPD in release 13 and we could continue the development of a common AOD/DPD format behind the scenes, taking special care to maintain user-level compatibility with the new SAN-based DPD and of course with the existing POOL-based AOD. The latter transition should be made easy by the T/P separation which will be completed before release 13.

6. Recommendations

Based on our findings and the discussion above, this task force recommends to

1st Do not replace POOL-based AOD with a ROOT-based one

Using the current SAN implementation, which is written by ROOT, as a replacement for the current AOD, which is by POOL, would require considerable core software efforts to implement and would constrain the ability to evolve the design and implementation (schema) of both AOD and SAN itself[1].

2nd Complete Transient/Persistent separation for current POOL-based AOD

While this release 13 deliverable is not strictly part of the mandate of this task force, we feel that it is an important complement of the short term fall-back solution discussed above. Given the current release schedule we recommend to schedule this deliverable for January 8.

1 This evaluation is based on the limited "schema evolution" support provided by the current ROOT version (5.x).

3rd Complete and put in production the current prototype of SAN-based DPD as an initial solution

As our evaluation shows, SAN is more convenient to use as DPD than existing AANT "flat" Ntuples. We recommend to put in production a SAN-based DPD for release 13[1]. This implies that after release 13 the standard DPD format will be SAN and not flat AANT. Together with the fully T/P separated POOL-based AOD, it will provide a foundation on which we can base any further development. This should also be a deliverable for January 8.

4th Develop an AOD/DPD "Grand Unification" format

We believe that the "SAN as pAOD" proposal described above will satisfy the "Grand Unification" goals discussed above. We recommend the development a functional prototype to be evaluated along the lines of our SAN-based and pAOD-based DPD evaluations described above. If our Task Force could perform an adequate evaluation and the results are positive, the prototype should be put in production, if at all possible in release 13. In such a case this recommendation supersedes our third recommendation.

7. Acknowledgments

We would like to thank for their contribution to the work of the Task Force: H. Ma, T. Maeno, S. Rajagopalan, and R.D. Schaffer.

1 This would include providing basic tools to write SAN-based DPD from an Athena job, and possibly one or more EventViews producing SAN as their output.

8. References

[SAN] https://twiki.cern.ch/twiki/bin/view/Atlas/AddingObjectsToAAN

[SAN AOD] http://indico.cern.ch/materialDisplay.py?contribId=1&materialId=slides&confId=5511

[CM] http://atlas-proj-computing-tdr.web.cern.ch/atlas-proj-computing-tdr/PDF/Computing-TDR-final-July04.pdf

[EV] https://twiki.cern.ch/twiki/bin/view/Atlas/EventView

[HighPT EV] https://twiki.cern.ch/twiki/bin/view/Atlas/HighPtView

[T/P] https://twiki.cern.ch/twiki/bin/view/Atlas/TransientPersistentSeparation

[Navigables] http://indico.cern.ch/getFile.py/access?contribId=s9t3&sessionId=s9&resId=1&materialId=0&confId=a036305

[BNL PAT meeting] http://indico.cern.ch/conferenceDisplay.py?confId=6529

[TPAOD]http://indico.cern.ch/getFile.py/access?contribId=s27t2&sessionId=s27&resId=0&materialId=0&confId=a057207

[pAODROOTAccess] https://twiki.cern.ch/twiki/bin/view/Atlas/RootAccessToPersistentAOD

Editor: Paolo Calafiura
Last modified: 18. Dec. 2006
First created: 10. Nov. 2006