A programmatic view of metadata, metadata services, and metadata flow in ATLAS

D Malon¹, S Albrand², E Gallas³, G Stewart⁴ for the ATLAS collaboration

¹ Argonne National Laboratory, Argonne, Illinois 60439, USA
² Laboratoire de Physique Subatomique et Corpusculaire, Université Joseph Fourier Grenoble 1, CNRS-IN2P3, INPG, 53 avenue des Martyrs, 38026, Grenoble, France
³ Department of Physics, Oxford University, Oxford, UK
⁴ CERN, Geneva, Switzerland

E-mail: malon@anl.gov

Abstract. The volume and diversity of metadata in an experiment of the size and scope of ATLAS are considerable. Even the definition of metadata may seem context-dependent: data that are primary for one purpose may be metadata for another. ATLAS metadata services must integrate and federate information from inhomogeneous sources and repositories, map metadata about logical or physics constructs to deployment and production constructs, provide a means to associate metadata at one level of granularity with processing or decision-making at another, offer a coherent and integrated view to physicists, and support both human use and programmatic access. In this paper we consider ATLAS metadata, metadata services, and metadata flow principally from the illustrative perspective of how disparate metadata are made available to executing jobs and, conversely, how metadata generated by such jobs are returned. We describe how metadata are read, how metadata are cached, and how metadata generated by jobs and the tasks of which they are a part are communicated, associated with data products, and preserved. We also discuss the principles that guide decision-making about metadata storage, replication, and access.

1. Introduction
In the execution of a single ATLAS [1] task, processing one input dataset to produce another, a non-trivial chain of metadata steps lies just beneath the surface, involving a wide range of metadata and components for metadata definition, storage, access, and processing. In this paper we provide an overview of the metadata and metadata flow involved in such processing, and indicate something of the design considerations and other issues involved even in an operation so apparently rudimentary. A detailed description of the metadata components involved is, however, beyond the scope of this paper. These components have for the most part been described elsewhere; for such information, see the references, and particularly [2].

Metadata are, formally, data about other data, but in practice what is meant by metadata may be context dependent. Trigger information and proton beam information, for example, while fundamental during proton collision runs, become metadata in the context of offline event analysis. While the scope of metadata may therefore be quite broad, our focus here is limited for practical reasons to semantic information used to understand ATLAS event data, the conditions under which they were produced or collected, and their further provenance, i.e., our concern here is for metadata directly relevant to physics purposes, an inventory of which may be found in [3]. Production system
metadata such as the sites on which jobs eventually ran and the time the task spent queued before running, accounting metadata such as the task submitter and data owner, and performance metadata such as per-algorithm execution time and output data size by object type, while worthy metadata topics in their own rights, are beyond the scope of this paper.

In the sections that follow we provide an overview of typical metadata steps in routine ATLAS processing, and a brief description of how metadata are returned. We examine in detail the important special case of event counting as an illustrative example, after which we turn to design considerations: the problem of mapping metadata about logical constructs onto physical ones, and some of the criteria that ATLAS has used to guide metadata design and deployment choices.

2. A simplified overview of metadata steps in processing

Even before a processing task is defined, it is likely that many metadata-related steps have been undertaken. Input dataset selection may have been based upon physics metadata, for which the principal ATLAS repository is AMI (ATLAS Metadata Interface; see [4]). Decisions about what to process and which auxiliary data to use to process it may have been made by consulting COMA (Conditions and configuration Metadata for ATLAS; see [5]), which provides an integrated view of configuration and conditions metadata, including, for example, trigger configurations and conditions data tags. Data quality decisions about which subsets of which runs are appropriate to use for physics purposes will have been made and reflected in so-called good run lists [6] [7].

In addition to decisions about which data to process, decisions must be made about which software versions to use and how to configure jobs to process the data. These decisions are encoded in a configuration tag that is recorded both in AMI and in the output event data products themselves. An example of a configuration tag is “r2713_p705.” The r and the p identify processing stages—reprocessing and filtering, broadly defined—and the digits that follow are akin to sequence numbers in the large range of job configurations ATLAS has used. These configuration identifiers are sufficient to allow determination of which ATLAS software releases, which conditions and geometry database releases, which input-to-output transforms, which component and algorithm configurations, and so on, were used in the processing steps used to produce the data from the original raw data input. The details corresponding to these configuration tags are recorded in the AMI database, and a service is provided to decode them. Importantly, ATLAS can configure, for example, a subsequent reconstruction transformation that corresponds to the way specific data have been produced simply by providing the relevant configuration tag as an input argument.

Once input data have been selected, software configured, and auxiliary data options defined, processing can begin. Job initialization is a metadata-intensive step, with algorithms and auxiliary services all requiring access to non-event data in order to initialize themselves to process event data appropriately. Some of this information comes from the configuration provided by the job transform itself or is specified by the accompanying configuration tag, but some will depend specifically upon the input dataset. Such information may in principle be dataset-level metadata that could be extracted exactly once from AMI at task definition time and propagated to all jobs. Typical current practice in ATLAS, though, is to “peek” into metadata contained in input files, possibly multiple times, first for the purpose of configuring an event reconstruction job, for example, and second for service and algorithm initialization within the reconstruction proper. This practice provides a safeguard against certain kinds of configuration errors—for example, errors that might arise when a physicist blindly reuses job options used for processing a different dataset—and the capability to configure jobs automatically in this data-aware way is quite useful, but it is also true that there are performance and consistency advantages to handling such configuration exactly once for all data within a dataset and not file by file or job by job.

Metadata such as detector conditions and calibrations that may vary over the course of a run may be accessed dynamically during event processing, either directly from databases or from caches derived and refreshed from such databases. There are input metadata transitions as well, arising, for example, when a job processes a sequence of input files. In this case the event loop and physics code do not care that a new file was opened, but certain auxiliary metadata may need to be updated or refreshed or flushed or accumulated or merged.
Certain metadata may be propagated from input to output, to be stored either in output data files or in metadata repositories such as AMI. Metadata may be created as well, and must be propagated in similar ways. Event counts, particularly when filtering is done, are a fundamental but important special case, worthy of separate treatment later in this paper. In any case, machinery must be provided to emit metadata and to transport them from the producing job through the production system infrastructure to appropriate repositories.

3. Returned metadata
Production transforms not only aggregate and propagate metadata to output data products, but they also return metadata for insertion into repositories for a variety of purposes. An elementary example is the list of files produced by a constituent job, their unique identifiers, and the datasets with which each of them is associated. That much would be required for correct task completion, but the files may also have semantic metadata associated with them, and such information should be returned as well. ATLAS has used a special-purpose output metadata file for such purposes with a format defined by an XML DTD, but has also used pickle files for certain information when metadata are returned in Tier 0 processing. Generally, physics metadata returned by jobs are routed unparsed into a production system database, from which they are retrieved and processed for insertion into AMI. In addition, an executing job may flag lines written into log files as containing information—{key, value} pairs, for example—to be included in the returned metadata. While this capability has been useful, metadata services are under development to provide a more explicit interface for this purpose.

4. Counting is fundamental
Event counting is fundamental. In many processing stages, event counting provides a straightforward integrity check, one of the most basic of which is whether the number of events in the output matches the number of events in the input. The question must be asked and answered, not only on a per-job level, but also on a task level, comparing input to output dataset event counts as a check that all relevant events in all relevant files have been processed. If a job filters its input, a process known as “skimming” events in high energy physics parlance, it remains important to confirm at least that it reads all of its intended input, and that it correctly records an output event count for downstream integrity checks. Production transforms routinely count events in output data products written in a wide range of data formats—and this is sometimes a challenge in itself, as physics groups may have their own particular and special output formats—and return event counts as semantic metadata associated with event output files.

In principle, because an event count is semantic information, as indeed is the fact that a given file contains event data at all, it is fair to ask whether an appropriately generic distributed data management system should care about such metadata, or whether such information should appear “only” in AMI, which is the ATLAS repository for physics metadata about datasets and their constituents. Event count, though, is important for task execution planning, because it is used to ascertain how many files should be input to each constituent job, or conversely how many jobs should be launched to process disjoint chunks of any single file, so there are pragmatic considerations to whether the collaboration should redundantly store event counts, say, in distributed data management catalogs. More on principles that guide ATLAS decision-making on such questions appears in a later section.

A subtler and more complicated counting requirement comes from the need, in most cases, to process complete luminosity blocks. Some background is in order here.

Many quantities relevant to event processing vary over time within a run. Conditions, calibrations, and machine luminosity are obvious examples; all are treated as essentially constant within a sub-run interval of time that ATLAS refers to as a luminosity block. Data quality assessments, too, are made at the granularity of a luminosity block. Certain changes to the trigger configuration, such as changes to prescale factors as luminosity decreases during the course of a run, may also occur on luminosity block boundaries, and event counting within the ATLAS multi-level trigger is also managed at the granularity of luminosity blocks.

For all of these reasons, it is essential that an analysis know whether it has seen all qualifying events within a luminosity block, for example to get the denominator right in a cross-section or other rate calculation. It is possible to add or drop complete luminosity blocks from an analysis and to handle
the resulting statistics correctly, but it is much more difficult when an analysis has seen only some of the events in a luminosity block, and the problem is particularly insidious if the physicist does not know that this has happened. ATLAS attempts wherever possible to store events from an integral number of luminosity blocks contiguously whenever possible. The metadata infrastructure propagates to downstream data products the list of complete luminosity blocks that have been seen by any given job. This is straightforward to accomplish when luminosity blocks are integrally contained in input, as long as jobs read every event in their input. The problem is slightly subtler when the event collection is the list of events returned by a query to the ATLAS TAG database. In this case, the list of luminosity blocks that constituted the domain of the query must be propagated, rather than the list built from pointed-to input files, and such propagation must be corrected if any files containing pointed-to events are unreachable or unreadable. When luminosity blocks are incomplete in any single file—as may happen with raw data from high-trigger-rate streams, for example, because the trigger writes multiple files in parallel and they may be too large for practical reasons to merge into a single file—much more careful accounting, including event counting, is required to ensure that complete luminosity blocks have been processed. While this is relatively straightforward in the controlled context of Tier 0 processing, where access to information regarding how many events the trigger system believes it has written is available, this requirement places demands downstream as well, for example on the merging of derived data files from multicore processing, which is another scenario in which multiple writers may have seen only a subset of a luminosity block even though that luminosity block may have been complete on input. Indeed, because merging metadata requires semantic knowledge not required in simple concatenation of arrays of events, metadata handling is often the principal source of complexity in efficient merging of event data files.

5. Logical versus physical constructs, and metadata mapping

Data-taking runs are logical, semantic units of data organization, as are contiguous temporal segments within a run, known as luminosity blocks. The same is true of the set of events that pass a particular trigger or suite of triggers. Files, on the other hand, are artifacts of storage organization. In most cases, a physicist needs to process a set of events corresponding to physics or data-taking criteria—for example, all the events from a given run that pass a given set of triggers—and whether those events are in one file or a thousand files, and whether those files also contain data from other events is, conceptually, a deployment detail. No resulting physics or physics metadata should depend upon storage organization. A natural implication might be that there should be no physics metadata associated with files, and in a sense this is true—in most cases, physics metadata about the file are really metadata about the collection of events within the file. This distinction might seem pedantic, but it turns out to be significant in a framework in which the I/O model is to process collections of events whether they are stored contiguously within a file or scattered across many files.

The ATLAS I/O infrastructure supports the notion of event collections much more directly than does the collaboration’s distributed data management infrastructure, and the I/O framework components and terminology reflect this [8] [9]. ATLAS event selectors iterate over event collections that may be explicit or implicit, with explicit meaning “this event list, or the set of events that satisfy this selection predicate” and implicit meaning “the set of events that happen to reside in this input file”—the latter only implicitly, at least from the point of view of data semantics, defining a collection.

Datasets provide something closer to a conceptual organization, but even here the physical mapping implemented by ATLAS and other experiments corresponds to exactly one logical view: in practice, a dataset is, either explicitly or effectively, a collection of files. For example, the file set containing Analysis Object Data (AOD) for a single run and trigger stream and a given processing is a dataset in this data management sense. In principle, though, the subset of events within this dataset that pass a specific set of triggers is conceptually every bit as logical a candidate for “dataset” status, and the fact that these events are mixed with events that passed other triggers that were written to the same stream is an artifact of storage. While processing such samples is routine, they are not currently supported as datasets per se by ATLAS distributed data management catalogs. Event collections instantiated as the output of queries to the TAG database, an event-level metadata facility [10] [11], are a surrogate for datasets of this sort and provide this functionality, as ATLAS software is designed to handle such event collections as input to ATLAS jobs both on grid and off [12] [13]. While there have been discussions about “virtual datasets”—datasets that are well defined but not physically instantiated as separate entities in storage—and how the collaboration might support a more extended dataset functionality not only in the I/O framework but in production data management as well, it is not
currently a common practice in ATLAS to do this. Such desiderata are currently being considered as input to the next generation of distributed data management software.

ATLAS employs a sophisticated in-file metadata infrastructure that goes far beyond machinery to allow storage of non-event data in event data files. This infrastructure has been described in earlier papers, and will not be discussed in detail here; see [14] for details. Worth noting here are the incident-handling features, as input file boundaries are artificial from the point of view of physics metadata, encountered asynchronously to state transitions of the event-processing framework—again, these artifacts of storage organization do not infiltrate ATLAS event and physics metadata handling. The architecture distinguishes the beginning and end of an input collection from the beginning and end of an input file even when those happen to be completely concurrent, because it supports the more general case of an event collection incarnated as a list of references to events that may reside amid other events in multiple files. In this case, the appropriate metadata to propagate to the output may not correspond to the complete metadata record of each input file, as only selected events are processed.

6. Design criteria and considerations

One of the principles guiding ATLAS metadata handling for production datasets is that if an event data file is unreachable, it must be possible to ascertain what is missing. Conversely, for routine use it should not be necessary to consult an external metadata source to understand a file’s content. The first statement ensures the integrity of an analysis when it must proceed without some relevant data—it must be possible to determine precisely what events are missing from an analysis, both to assess the significance of the omission and to correctly adjust statistics and cross-section calculations accordingly. The second ensures that all analysis needn’t cease when a physicist is on an airplane, or when a network connection to a remote database server is down.

While these principles sound straightforward enough, the first has stringent implications for bookkeeping, and for luminosity-block-level accounting and management. A related consequence is that ATLAS takes pains to avoid splitting luminosity blocks across file boundaries whenever possible, and scrupulously accounts for the cases in which this cannot be accomplished. The converse half of the principle guides decisions about what subset of metadata are stored in event data files, and following this principle is what allows file peeking utilities to support correct configuration of jobs that read ATLAS data.

Once the capability to store metadata in files has been provided, there is a temptation for developers to use it as a bucket for anything that might prove useful, so that no downstream client ever requires access to an external data source. Principles that have been applied as criteria to decisions about in-file metadata content include mutability, pervasiveness, and frequency. The mutability criterion reflects the fact that if metadata are likely to change, then they may not be good candidates for caching within event data files because of the risk that physicists may use outmoded values unwittingly. Pervasiveness refers to how widespread the need for the metadata might be—if they are needed only rarely or by a few analyses, caching may not be called for or worth the additional storage costs. Frequency refers to the number of times the information will be accessed. If every analysis needs the information, but only once at the point at which their samples are being defined, then the requirement is pervasive but not frequent. If the information is needed every time the file is opened but only for exceptional specialist purposes, then the requirement is frequent but not pervasive. While these principles have guided decisions about in-file metadata in particular, they are also relevant to decisions about auxiliary data distributed with ATLAS releases or made available in caches of database-resident data.

Storing equivalent metadata in multiple places, while sometimes convenient, poses certain risks. In some cases, the question of whether a metadata instance is redundant is ambiguous—a view of metadata extracted from another source, for example, is arguably redundant, but if the view is time-consuming or expensive to repeatedly reproduce or access, then it may be worthwhile to maintain the derived view as separate metadata. ATLAS decisions regarding metadata redundancy are guided principally by these pragmatic concerns, but a design principle is that even in these cases, it must always be clear what the authoritative source or repository of information is.
7. Conclusions and future work

Metadata are integral to every aspect of ATLAS computing. The intent of this paper has been to provide an illustrative view of ATLAS metadata, principally from the point of view of the infrastructure and services needed for metadata flow in the context of a single task. While metadata components and infrastructure have grown organically as the experiment has matured, a number of principles described herein have informed their design and connectivity. The infrastructure continues to evolve in a variety of ways, with improvements planned to how dataset-level metadata may be used to reduce the need for peeking into input files, to how metadata are emitted and transported from executing jobs to the collaboration's metadata repositories, and to machinery for robust accounting of low-rate error conditions in physics data bookkeeping.

Acknowledgments

Argonne National Laboratory's work was supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under contract DE-AC02-06CH11357.

References

[1] The ATLAS Collaboration 2008 The ATLAS experiment at the Large Hadron Collider JINST 3 S08003
[2] Gallas E, Malon D, Hawkings R, Albrand S, and Torrence E 2010 An integrated overview of metadata in ATLAS J. Phys.: Conf. Ser. 219 042009
[3] Costanzo D et al 2007 Metadata for ATLAS CERN Report ATL-GEN-PUB-2007-001
[4] Albrand S, Fulachier J and Lambert F 2010 The ATLAS metadata interface J. Phys.: Conf. Ser. 219 042030
[5] Gallas E 2012 Conditions and configuration metadata for the ATLAS experiment Preprint, these proceedings
[6] http://atlas-runquery.cern.ch/
[7] Adelman J et al 2010 ATLAS offline data quality monitoring J. Phys.: Conf. Ser. 219 042018
[8] van Gemmeren P and Malon D 2009 The event data store and I/O framework for the ATLAS experiment at the Large Hadron Collider IEEE Int. Conf. on Cluster Computing and Workshops 2009 1
[9] Malon D, van Gemmeren P, Cranshaw J and Schaffer A 2006 Sailing the petabyte sea: navigational infrastructure in the ATLAS event store Proc. Computing in High Energy and Nuclear Physics 2006 (Mumbai: Macmillan) 312-5
[10] Cranshaw J et al 2008 Building a scalable event-level metadata service for ATLAS J. Phys.: Conf. Ser. 119 072012
[11] Malon D, Cranshaw J and Karr K 2006 A flexible, distributed, event-level metadata system for ATLAS Proc. Computing in High Energy and Nuclear Physics 2006 (Mumbai: Macmillan) 316-9
[12] Cranshaw J et al 2010 Event selection services in ATLAS J. Phys.: Conf. Series 219 042007
[13] Doherty T et al 2012 TAG-based skimming in ATLAS Preprint, these proceedings
[14] Malon D et al 2008 An inconvenient truth: file-level metadata and in-file metadata caching in the (file-agnostic) ATLAS event store J. Phys.: Conf. Ser. 119 042022