ATLAS

ATLAS Metadata Task Force

Metadata for ATLAS

Document Version: 2

Document ID:

Document Date: 2007-04-04

Document Status: Released

Abstract

This document provides an overview of the metadata, which are needed to characterize ATLAS event data at different levels (a complete run, data streams within a run, luminosity blocks within a run, individual events).

Authors:

D. Costanzo, J. Cranshaw, S. Gadomski, S. Jezequel, A. Klimentov, G. Lehmann Miotto, D. Malon, G. Mornacchi, P. Nemethy, T. Pauly, H. von der Schmitt

Ex officio: D. Barberis, F. Gianotti, I. Hinchliffe, L. Mapelli, D. Quarrie, S. Stapnes

1. INTRODUCTION 4

1.1. Purpose of the document 4

1.2. Glossary and Abbreviations 5

2. METADATA ORGANIZATION 8

3. EVENT-LEVEL METADATA 8

4. METADATA FOR FILES 8

4.1. Index Files Created Online 8

4.2. File Metadata for Offline Analysis 9

5. METADATA FOR LUMINOSITY BLOCKS 9

5.1. Luminosity Block Metadata Generated Online 9

5.2. Examples of Detector Status Changes 12

5.3. Usage of Luminosity Blocks in Recording Detector Status 13

5.4. Timeline for Production of Detector Status Metadata 14

5.5. Offline Information 15

6. METADATA FOR RUNS 15

7. METADATA FOR DATA STREAMS 18

8. METADATA FOR DATASETS 19

8.1. Dataset Definition 19

8.2. Dataset Identification and Naming 19

8.3. Dataset Composition, Content and Description 20

8.4. Dataset Management Information 20

8.5. Project Information 20

9. ANALYSIS USE CASES 23

9.1. Calibration and Cross-section Measurement of Inclusive Z-> ee Production 23

9.2. A Data File is Inaccessible or Unreadable 24

10. ACKNOWLEDGEMENTS 24

11. REFERENCES 24

Table 1 Document Change Record

Title: Metadata for ATLAS			
ID:			
Version	**Issue**	**Date**	**Comment**
1	1	21 Jul 06	Initial Version
2	1	5 Feb 07	Draft for Comments released

1. Introduction

The ATLAS experiment is going to produce a large amount of data, both during online data taking and during offline processing of the events. In order to be able to analyse them, information is needed which goes beyond the content of the data itself. In addition, all those data need to be catalogued in a way, which will allow retrieving interesting subsets without having to read the complete data content. Thus, metadata need to be defined, which describe the main features of data in a concise way.

1.1. Purpose of the document

Metadata are, strictly speaking, all data which describe other data, ranging from raw event headers to configuration settings (detectors, DAQ and Trigger). This document does not aim at reviewing the content and format of all metadata: in particular, we exclude any detailed description of system specific parameters (e.g. the trigger configuration, DCS settings for individual sub-detectors). Where this information is needed for data characterization we will consider as metadata a pointer to the detailed information (e.g. DAQ configuration version). Similarly, we will not review the raw event format and the content of the headers therein [1], nor the information on events (event tag) which has been recently worked out by the Event Tag Task Force [2].

This document describes the metadata, which have to be created at different granularity levels in order to characterize event data. Order of magnitude estimates of the differences in scale in the number of events contained are given below.

- Event=1
- File=10^2-10^4
- Luminosity block =10^4
- Run=10^5-10^6
- Data Stream per run=10^4-10^6
- Dataset=10^5-10^9

Some of these scales are due to hierarchical arrangements in how the data is taken: run, luminosity block, event. Others are obviously due to arbitrary decisions of users of the event store: file, dataset. Data stream falls in between and will hopefully be better defined in the report from the Streaming Working Group [4]. Also, in the current planned run structure [5], the identifiers for luminosity block and event are only defined within the context of the associated run, i.e. they are only unique within that run.

A means to navigate between these ‘granularities’ of scale will be needed to support queries where the user indicates a requirement on a given granularity but where the data needed is stored in a different granularity, e.g. ‘Give me the luminosity for those runs where all of the detector elements are marked as good that went into Stream A’.

Metadata will be used in a variety of ways.

- To determine the status of a process such as calibration or analysis (e.g. stage of processing).
- To provide statistics on some process without having to read in large amounts of data (e.g. run completed gracefully or terminated abnormally).
- To provide early indications of problems and clear indications of their source (e.g. detector status summary).
- To allow the cross referencing of data in different data stores or different formats (conditions, events, TDAQ), (POOL, n-tuple), …
- To locate other types of data.
- To encapsulate the state of a process so that it can be repeated or easily modified (e.g. versions of software and configuration used for a process).

In section 3 we list metadata, which characterize individual events. Section 4 describes metadata needed for data files. In section 5 we make an inventory of metadata, which describe events of a same luminosity block within a run and section 6 contains the metadata, which bond events of the same run. Finally, sections 7 and 8 describe the information needed at stream and dataset levels. Each section is further subdivided into different sub-sections, which detail out what metadata will be produced at what stage (e.g. online, offline). For every entry we determine where the information comes from and where it has to be finally stored.

Section 9 contains a few use cases that have been used to validate the metadata organization.

1.2. Glossary and Abbreviations

ACL	Access Control List
AMI	ATLAS Metadata Interface
AOD	Analysis Object Data
ATLAS	A Toroidal LHC ApparatuS
CERN	European Laboratory for Particle Physics
CTP	Central Trigger Processor
DAQ	Data AcQuisition System
DB	Database
DCS	Detector Control Systems
DDM	Distributed Data Management
DFM	Data Flow Manager: component of the DAQ software which manages the event building process.
DIP	Data Interchange Protocol
ECR	Event Counter Reset: signal regularly sent to the complete front-end to reset the counter of level 1 accepted events. It is used to resynchronize the L1ID across ATLAS.
EF	Event Filter (third level trigger): farm of processors which run "offline-like" algorithms to select the events which will be put on mass storage.
EOR	End Of Run
ESD	Event Summary Data
GID	Global event ID: the GID is a 32 bit integer that uniquely identifies an event during a run. It is associated to the raw data during the event building phase, i.e. after the second level trigger. Thus, it is not time ordered.
GPS	Global Positioning System
GUI	Graphical User Interface
ID	Identification
Inefficiency (of bad)	Should estimate the probability that a tracking detector or calorimeter does not find an actual track or shower, respectively, because of bad channels. The inefficiency is definitely not equal to the fraction of bad channels F_{BAD}, but should be a monotone nonlinear function of F_{BAD}, growing only slowly for a large range of F_{BAD}, but rising sharply to 1.0 when F_{BAD} approaches 100%. Inventing this in detail will take much (detector dependent) development.

IS

Information Service: component of the DAQ software which allows distributing and retrieving (formatted) information between processes.

L1ID

Level 1 ID: the L1ID is a 24 bit integer that is assigned to each event accepted by the first level trigger. It is strictly time-ordered, thus it is equivalent to a time stamp for the event. Often (always in this document) term L1ID is used to indicate the extended L1ID: this is a 32 bit integer which contains the L1ID in the lower 24 bits and the ECR counter in the 8 highest bits. The extended L1ID is not expected to be unique within a run.

L2PU

Level 2 Processing Unit: component of the TDAQ which executes the level 2 trigger algorithms.

L2SV

Level 2 SuperVisor: component of the DAQ which manages the load balancing within the farm of L2PUs.

LB	Luminosity Block
LBN	Luminosity Block Number
LHC	Large Hadron Collider
PT	Processing Task: component of the EF which is in charge of algorithm execution.
RDO	Raw Data Object
ROS	Read Out System
S32	Signed 32-bit integer
SCN	Stable Condition Number

SFI

Sub-Farm Input: component of the DAQ in charge of performing event building. In ATLAS there will be ~ 100 SFIs.

SFO

Sub-Farm Output: component of the DAQ in charge of putting the event accepted by the third level selection to mass storage. In ATLAS there will be ~ 10 SFOs.

SOR	Start Of Run
SWIG	SoftWare Integration Group
TAP	Triggers After Prescale
TAV	Triggers After Veto
TBP	Triggers Before Prescale
TBV	Triggers Before Veto
TDAQ	Trigger and Data AcQuisition

Thrust (of bad)

Should characterize how isotropically (i.e. randomly) versus how directionally the bad channels of a given detector are distributed. The jet physics definition of Thrust T and thrust axis $\hat{n}_T$ is $T = \max\left(\frac{\sum|\vec{p}_i \cdot \hat{n}_T|}{\sum|\vec{p}_i|}\right)$, with $\hat{n}_T$ chosen to maximize T. It is a momentum-weighted average, designed to be forward-backward symmetric along the thrust axis. For our detector characterization, we do want forward-backward information, so we remove the absolute value; we also want all detector elements to have the same weight, so we replace $\vec{p}_i$ by $\hat{n}_i$, a unit-vector in the direction of the bad elements. Then a calculation of thrust reduces to finding the vector sum $\vec{V} = \frac{1}{N(bad)}\sum_i \hat{n}_i$ with $T = |\vec{V}|$ and $\hat{n}_T = \frac{\vec{V}}{|\vec{V}|}$. The limits are $T \cong 1.0$ for bad elements all clustered in a given direction $\hat{n}_T$ and $T \cong 0$ for the bad elements in random directions.

U32	Unsigned 32-bit integer
VO	Virtual Organization
VUID	Version Unique ID

2. Metadata Organization

Metadata are created at very different levels, before taking the data, during a run, at the end of it and throughout different stages of offline processing.

Each system should provide sufficient metadata information to allow the next level to select the relevant data and be able to process them.

This report does not address specifically implementation and organization of the metadata storage. Nevertheless, for clarity, the suggested destination of metadata is indicated, when known. For several parameters the conditions DB is the natural place to store information; this is particularly valid for the run parameters and LHC parameters described in section 6, as well as for detector status changes (see section 5.2). It is not clear at this stage, where statistical information, such as event counts, scalers' information, size, location and name of raw data files, etc should be stored. They are not really "conditions". On the other hand at present there is no other database (accessible from the DAQ) for recording those.

For offline datasets, metadata can be subdivided in properties used internally for DDM and those, which need to be exposed to users via the AMI interface.

3. Event-level Metadata

In this document we will not address the event level metadata. They are described in [2].

4. Metadata for Files

The ATLAS Luminosity Task Force Report [3] makes three specific recommendations regarding file-level metadata:

[Recommendation 6.3]: The ATLAS metadata system must be able to determine the luminosity blocks and trigger chains associated with a given production data file or event collection. This capability must be external to the files or collections themselves, i.e., one needs to be able to obtain this information even when the files themselves are unreachable or unreadable.

[Recommendation 6.4]: The ATLAS metadata system should, given luminosity block and trigger chain information, be able to determine which production data files and event collections at a given processing stage contain events from that block and trigger chain. This is the converse of the previous recommendation.

[Recommendation 6.5]: As ATLAS begins to address the issue of provenance tracking and what a file or event collection internally "knows" about its contents, history information regarding the luminosity blocks used as input to its production should be included in any provenance record associated with or contained within that collection or file.

4.1. Index Files Created Online

Raw data files written by the DAQ contain some metadata, documented in [7]. In addition to this the SFO application will produce so called "index files". Each data file will have a corresponding index file. The index file will be a human-readable ASCII file containing one line per event. The line will contain:

- event number,
- a summary of the trigger information,
- the stream tag,
- offset of the event in the data file.

The main purpose of the index files is to provide information in case the data files go missing. For this reason the index files will be written on different disks, physically installed in different PCs, with respect to the data files. The index files will in principle become redundant after the offline processing reads the raw data files. One can archive the index files for more redundancy.

In addition to the index files, the SFO will also produce a list of all the data file names, as mentioned in section 7. The run number and the luminosity block number are part of the file name. Therefore, if a data file goes missing, it will be possible to know, from this list, which one it is and what data it contained.

Issue for follow up: It is still unclear where the event index files as well as the list of datafiles written by the SFOs shall be stored.

4.2. File Metadata for Offline Analysis

File-level metadata must include:

- The run and luminosity block ranges used as input into the file's production (ranges of integer pairs);
- The list of input trigger chains (which may be summarized by stream information in early stages, though a specific subset of triggers may be used for filtering a stream in later offline processing stages).

Note that, for offline analysis, one must retain information about the *input* run and luminosity block ranges, because this information will be needed for cross-section calculation even if no events from certain input luminosity blocks survive a cut and become part of the output event list.

5. Metadata for Luminosity Blocks

In this section we will consider the metadata, which are needed to characterize events, which are taken within the same luminosity block in a run (identified by a run number and luminosity block number).

The first source of metadata is provided by the TDAQ system online, at the start of each luminosity block and at the end of it. For safety reasons all essential information must be stored at the start, while some more data can be added for book-keeping and statistics (trigger scalers, natural end of block or end forced by errors,…) at the end.

Additional metadata will be added for each luminosity block offline, as soon as the raw data are processed and when other information (e.g. run control, DCS, …) is used to complement the data characterization.

5.1. Luminosity Block Metadata Generated Online

The DAQ and DCS systems will use the Information Service to make (time stamped) metadata publicly available. At present there is a software application, which can extract selectively data from the Information Service and insert them to the conditions DB.

The metadata that can be made available by the TDAQ system at the start of a new luminosity block are listed in Table 2 while the metadata available at the end of a luminosity block are listed in Table 3.

Issue for follow up: it is still unclear in which format the Run Control shall list the newly enabled/disabled channels. This might be done as a series of channel identifiers or, more probably, by storing the new version of the configuration DB in use (in this case the version number will be put as metadata).

Issue for follow up: the format and detail of the luminosity estimates to be included as metadata online is not defined yet.

Table 2 Metadata inserted by the online system at the beginning of a luminosity block.

Name	Type	Scope	Origin	Destination
Luminosity Block Number	16 bits	Key which uniquely identifies a luminosity block within a run	Run Control	Raw data Conditions DB
Disabled /Re-enabled Channels	?	List of newly Disabled/Enabled DAQ channels (ROD Ids, ROB Ids, ROS Ids)	Run Control	Conditions DB Raw Data
L1 pre-scale settings	256 x U32	LVL1 pre-scale settings	Trigger Configuration (CTP)	Conditions DB
CTP start time	2 U32	GPS time-stamp at start of luminosity block	CTP	Conditions DB
Start L1ID	U32	L1ID at start of luminosity block	CTP (needs to be taken during busy)	Conditions DB

Table 3 Metadata inserted by the online system at the end of a luminosity block.

Name	Type	Scope	Origin	Destination
CTP end time	2 U32	GPS time-stamp at end of luminosity block	CTP	Conditions DB
End L1ID	U32	L1ID at end of luminosity block	CTP (needs to be taken during busy)	Conditions DB
L1 accepts/rejects	2 x U32	Number of events accepted/rejected	CTP	Conditions DB
L1 TBP	256 x U32	Number of Level-1 triggers before pre-scale per trigger item	CTP	Conditions DB
L1 TBV (=TAP)	256 x U32	Number of Level-1 triggers before veto (= after pre-scale) per trigger item	CTP	Conditions DB
L1 TAV	256 x U32	Number of Level-1 triggers after veto per trigger item	CTP	Conditions DB
LHC turns	U32	Number of LHC turns per LB	CTP	Conditions DB
L2 accepts/rejects	3 x 2 x U32	Number of L2 accepts/rejects	Sum over L2SV, L2PU counters and DFM	Conditions DB
L2 accepts before and after pre-scale per trigger chain	2 x #trigger chains x U32	Number of L2 triggers before and after pre-scale for pre-scale corrections	Sum over all L2PU	Conditions DB
Events in SFI	U32	Number of events in SFI	Sum over SFIs	Conditions DB
EF accepts/rejects	2x U32	Number of EF accepts/rejects	Sum over EF PTs	Conditions DB
EF accepts before and after pre-scale	# trigger chains x 2 x U32	Number of EF accepts before and after pre-scale per trigger chain	Sum over EF PTs	Conditions DB
Events in SFO	U32	Number of events in SFO	Sum over SFOs	Conditions DB
Luminosity estimates	Either 1 Double or 3564 Double per method	Luminosity estimates of various methods	Luminosity monitoring applications, DCS	Conditions DB

5.2. Examples of Detector Status Changes

Figure 1 gives an example of several events changing detector status. In practice, of course, the status will not change on every other LB but much less frequently. From top to bottom are shown:

A readout problem (crash of a ROS) is detected and handled synchronously by DAQ. As a consequence, part of the readout chain is disabled, and the CTP is informed to terminate the present luminosity block, LB b, and to start LB b+1. Triggers resume only after that. (Note that to our present understanding the inverse event, restoration of the part of the readout in question, does not happen automatically during a run.)

A failure in detector services (trip of a HV supply) is detected by DCS. It is reported to run control and to the conditions database. Both happen with some delay, O(msec-sec), due to the finite time it takes to measure the status change, and due to the finite time resolution of DCS and of the computing services involved. The status change is reported during LB b-1. Usually the run resumes, as in the example. The LB number does not change.

The HV failure is restored. This is reported in LB b+2 with the delays mentioned. The LB number does not change.

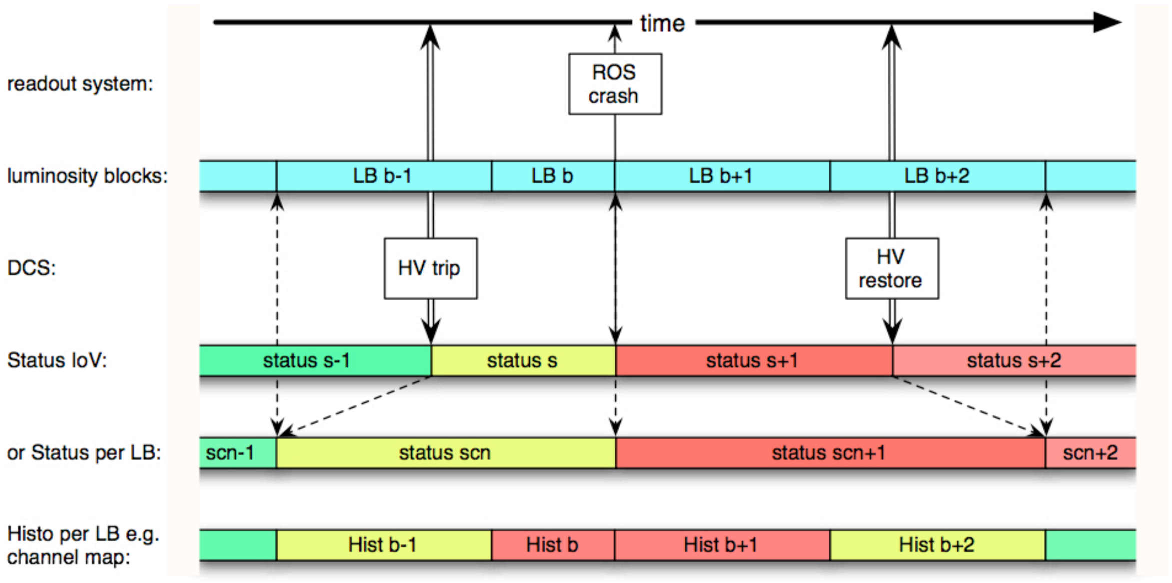
The situation of the example can be recorded as a (composite) status over time. “Status IoV” changes from green (status s-1) to yellow (status s) when the HV trips, to red (s+1) when the ROS crashes (a part of the detector will not be read out for the rest of this run). Then it becomes slightly less red (s+2) when the HV is restored. This makes sense if the readout problem and the HV problem affect different parts of ATLAS. The composite status would comprise traffic lights per sub-detector with some granularity, e.g. distinguishing barrel, end caps, etc. The composite status could be enhanced to contain per-purpose traffic lights, e.g. “good for missing Et”, etc.

The IoV based status can also be quantized to LB boundaries as “Status LB”. In this case, changes to the worse would be advanced to the start of the LB where the change happens, and improvements would be retarded to the end of that LB block. The readout problem coincides precisely with a LB change and is neither retarded nor advanced. – The quantized status intervals are numbered with SCN (stable conditions number), see next paragraph.

There are other sources of status information, e.g. data quality histograms. Some of them will be taken per LB. Consider a channel map which covers the channels affected by the HV trip – the histogram will be partly bad (yellow), i.e. missing part of the nominal entries, during LB b-1 when the trip occurs and during LB b+2 when HV is restored. During LBs b and b+1 the histogram is bad (red). Note that this histogram does not add to the knowledge we have already from DCS, and that the detector status must not be set worse than it is on the basis of DCS. But there may also be new, additional information in some histograms which is not visible from DCS.

Status changes due to readout problems would be recorded online. DCS status could be recorded online, although with some difficulty because correlations with other pieces of information may be needed. We envisage the status from DCS to be assessed after data taking, but before first pass reconstruction, in an offline job scanning the conditions information archived by DCS. Finally, status information from data quality histograms would be recorded offline.

Figure 1 Example of events modifying the detector status as a function of time.



5.3. Usage of Luminosity Blocks in Recording Detector Status

Luminosity blocks can serve as a quantum for the detector status. Its metadata is organized as ‘bad-lists’ in the Conditions DB for each detector, ordered by the time-stamp of changes. At start of run, the original bad-list is used, to which newly-bad lists are added every time there is a change in detector status.

Suppose we number the stable conditions between changes with a **Stable Condition Number SCN** 0,1,2, ..., where 0 corresponds to the run start condition. After each change the starting and ending luminosity block number LBN for that detector condition can be determined from the two time-stamps and stored.

We end up with a list (separately for each detector) with:

SCN	Start time	stop time	run number	start LBN	stop LBN
0					
1					
2					
3					

We note the fact that the SCN table is not a database schema but an abstract table and logic representation, which will be mapped onto whatever DB implementation is practical. Only once after each change (i.e. once per SCN), can quantities be calculated that need to be obtained from the bad list: number of bad units, “traffic light color”, and others. It is useful to calculate along with the number of bad units the corresponding inefficiency (see definition in section 1.2), the number of newly-bad and newly-restored units, and thrust (see definition in section 1.2). Following [4], we call N_{BAD} the number of bad units, $\Delta = N_{BAD}(new) - N_{BAD}(old)$ the change in the number of bad units and $\Delta' = N_{BAD}(old) - N_{BAD}(new)$ the number of newly-restored units.

Concerning inefficiencies, it is also useful to distinguish estimated inefficiencies from Nbad at two different granularities, coarse grain where the unit could be a sector or module (for instance a HV trip) and fine grain from single bad channels. Finally, an inefficiency may have nothing to do with bad channels, but may affect the whole detector (for instance an electron-eating gas impurity in a gaseous detector). These three inefficiencies can be defined and tracked separately, in a mutually exclusive way.

These “shorthand results” are the only detector status information added to the luminosity block metadata for each LBN, together with the SCN value for that LBN (from the list above). Then the luminosity metadata (stored in the conditions database) will contain, in addition to luminosity info, the “shorthand” detector conditions, for each detector, as shown in Table 4.

Table 4 Summary of detector status for a luminosity block.

Name	Type	Scope	Origin	Destination
SCN	?	Pointer to the right bad lists	Calculation after run and before Tier0	Conditions DB per LBN
Transition Flag	Boolean	Changing conditions during luminosity block	Calculation after run and before Tier0	Conditions DB per LBN
Traffic Light	UInt	Red, Yellow, Green for overall detector condition	Calculation after run and before Tier0	Conditions DB per LBN
Global inefficiency	?	Global inefficiency	Calculation after run and before Tier0	Conditions DB per LBN
Coarse grain (modules or sectors)	?	N_{BAD}, $\Delta(N_{BAD})$, Thrust(of bad), Ineff(of bad)	Calculation after run and before Tier0	Conditions DB per LBN
Fine grain (channels)	?	N_{BAD}, Δ (N_{BAD}), Thrust(of bad) Ineff(of bad)	Calculation after run and before Tier0	Conditions DB per LBN

These quantities should be calculated and mapped onto luminosity blocks after the run has ended, but before the Tier-0 reconstruction, for two reasons: Firstly, these calculations are to a large extent independent of the event data, and secondly, Tier-0 reconstruction might use some of the results, such as the traffic light colour.

Out of the shorthand detector status, the traffic light will also go to the individual event record as event metadata, as specified in [2].

In addition to being added to the luminosity block information, the shorthand table is also kept with each entry in the SCN table, since it is common to the whole entry.

Issue for follow up: The content of the conditions to be stored for each detector has to be specified by the different detector groups. The way in which this information is used to create the summary of the detector status and, in particular, the “traffic light” entries in the metadata has to be discussed within the Data Quality activity.

5.4. Timeline for Production of Detector Status Metadata

STEP 0. At start of run: Initial Detector Bad-lists for individual sub-detectors are saved in the Conditions Data Base.

STEP 1. Real-time or Near-real-time during a run: New individual sub-detector troubles saved by the sub-detector groups as time-stamped detector-bad-list-updates in the Conditions Data Base.

STEP 2. At end of run: bad-lists revised, if necessary from examination of on-line monitoring histograms. [Final bad list is initial bad list for the next run].

STEP 3. After end of run and following any bad-list revisions: **Stable Condition Time Table** is created.

STEP 4. After run end and before Tier-0 reconstruction: An automated process (script) is run to calculate all the detector summary data for each sub-detector and for each line in the **Stable Condition Time Table.** The detector summary information is saved in the Conditions Data Base, located with and linked with the **Stable Condition Time Table.**

STEP 5. After STEP 4 and before Tier-0 reconstruction. Detector summary information for each sub-detector is added to the luminosity-block-information record for each luminosity block.

STEP 6. During Tier-0 reconstruction: add the Traffic-light-colour to the event metadata for each sub-detector and each event.

STEP 7. Any time during off-line analysis, and only if necessary: Revise detector bad list, **Stable Condition Time Table,** and detector summary information if off-line analysis indicates troubles not previously discovered. Flag run for repeating the off-line analysis with the new conditions incorporated.

In general, the procedure to move from the online acquired raw data and detector and trigger conditions, to the offline reconstruction will be quite complex in ATLAS. This topic is only indirectly related to the scope of this document and is presently being addressed by the Data Preparation and Data Quality activities. Nevertheless, information has been gathered from the CDF and the D0 to see how they handle this transition. A summary of it can be found in appendix A.

5.5. Offline Information

Offline processing at the Tier-0 and beyond will take place respecting the luminosity block boundaries. Quality information will be produced for detector status and also for the quality (or reliability) of the outcome of calibration/alignment and reconstruction procedures. This information, whose details are currently being discussed by the Data Quality group, will be stored in the Conditions DB.

The luminosity for each block will also be calculated offline, using machine information, detector measurements, and physics processes as input. All versions of the individual measurements, as well as the best estimate of the luminosity for each block, will be stored in the Conditions DB and made available for physics analysis.

Analysts at the beginning of their event selections will have to decide, on the basis of DQ information, which luminosity blocks have to be used as input for their analysis; at the end of the analysis, the same list of luminosity blocks must be used to evaluate the integrated luminosity for that data sample.

6. Metadata for Runs

A run is essentially an online concept that divides data according to main detector or DAQ conditions. Any metadata needed to describe these conditions are therefore considered to be run level metadata. Runs are identified by a unique run number.

The first source of metadata is provided by the TDAQ and DCS systems online. Additional metadata might be added for a Run offline, as soon as the raw data are processed and other information (e.g. from LHC machine, DCS, calibration constants, etc) is used to complement the data characterization. At the moment we haven't identified offline metadata, which would be created at Run level.

The DAQ and DCS will create and store run level metadata at the beginning and at the end of each run.

For safety reasons all essential information must be stored at run start, while at run stop some more data can be added for book-keeping and statistics (time of run end, clean/unclean run stop, etc).

The DAQ and DCS systems will use the Information Service to make (time stamped) metadata publicly available. At present there is a software application, which can extract selectively data from the Information Service and insert them to the conditions DB. Additionally, some of the metadata will also be stored in each raw data file.

At the start of a new run the metadata listed in Table 5 can be made available by the TDAQ system.

At present there are additional metadata used by individual detectors specifically for commissioning, calibration, etc. Their name and possible values are specified in the DAQ configuration DB; their value for a run is chosen by the shifter from the GUI: these are automatically inserted into the Online Information service and, from there, they can be stored into the Conditions DB and also into the raw data files. We suggest keeping those freely settable by the sub-detectors for their needs.

At the end of a run the TDAQ will set additional metadata, as listed in Table 6.

Issue for follow up: the detailed type of information to store the DAQ and trigger configuration version is not specified yet.

Issue for follow up: it shall be investigated whether the conditions DB is the right place so store statistical information (e.g. number of triggers taken, raw data files written, etc) at luminosity block, run and data stream level.

Table 5 Metadata inserted by the online system at the start of a run.

Name	Type	Scope	Origin	Destination
Run Number	31 bits	Key which uniquely identifies a run	Run Number Server	Raw data file Conditions DB
SOR Time	Time	Allows association of time stamps and run number	Run Control	Raw data file Conditions DB
Run Type	String/enum	Human readable description of run (Calibration, physics, ...)	At present possible options reside in Configuration DB; type chosen by shifter via GUI	Raw data file Conditions DB
Detector Configuration	?	Key to get DAQ Configuration DB version	Configuration DB	Conditions DB
Trigger Configuration	?	Key to get to trigger settings saved in a database	Trigger Configuration DB	Conditions DB
Filename_tag (useful during commissioning, might be automatically generated for final ATLAS)	String	Used to build up the file name of raw data files.	At present set by shifter via GUI.	Raw data file Conditions DB
Data Source	String	Human readable string determining the source of data: LHC beams: proton, ion, empty; cosmics, radioactive source, pulses,…)	directly from the LHC (via DIP); in absence of beam type chosen by shifter via GUI	Raw data file Conditions DB
LHC Fill Number	U32	Fill number of the accelerator	Directly from LHC via DIP.	Conditions DB
Beam momenta	2x S32	Momenta of beam 1 and 2	Directly from LHC via DIP.	Conditions DB
Filling scheme, number of bunches	string, enum	Filling pattern of the beams. Could be a version number of a filling scheme, or simply the number of bunches	Directly from LHC via DIP.	Conditions DB

Table 6 Metadata inserted by the online system at the end of a run.

Name	Type	Scope	Origin	Destination
EOR Time	Time	Allows association of time stamps and run number	Run Control	Conditions DB
Total Time	U32	Statistics	Run Control	Conditions DB??
Clean Stop	Boolean	Statistics	Run Control	Conditions DB??

7. Metadata for Data Streams

Table 7 lists all the metadata needed to describe a stream of raw data. We concentrate here on the data produced online, by the TDAQ system.

In the offline processing the data stream becomes one example of a dataset. Attributes of (raw data) datasets will need to be sufficient to contain the information created by the online system, as well as further additions by the offline.

The last two items, i.e. the list of data files and the number of events will be changing during the run and will have final values only after the run is stopped. The other parameters of streams will be available at the beginning of the run.

Table 7 Metadata inserted by the online system for each data stream and per run.

Name	Type	Scope	Origin	Destination
Short name	string	Identifies raw data files. Is part of the file names.	Trigger and DAQ Configuration	Conditions DB
Descriptive name	string	human readable and meaningful physics name (e.g. “electron and gamma”)	Trigger Configuration	Conditions DB
Trigger Menu	?	List of keys (?) to trigger chains associated with this stream	Trigger Configuration	Conditions DB
Respecting Luminosity Blocks	bool	Set to ‘true’ if this stream has each file closed at the end of a luminosity block	DAQ Configuration	Conditions DB
List of data files	list of strings	Names of all the data files that have been written for this stream by all the SFOs.	SFO applications	An ASCII file
Number of events	usigned int	Number of events written into this stream	SFO applications, but numbers have to be added	Conditions DB

8. Metadata for Datasets

8.1. Dataset Definition

Raw data goes through several stages of offline processing before it is in a format useful for a physics analysis. The parameters and the configuration used during this offline processing need to be saved in the metadata. As an example, the software release used for reconstruction needs to be recorded, as the offline quality of the reconstructed objects depends on the version of the software used.

Events are grouped into datasets initially during the streaming phase at the Tier-0, but subsequently offline selection criteria can be used to create datasets. From an operation viewpoint, datasets are saved on storage elements distributed on the GRID and handled by the ATLAS Distributed Data Management (DDM) system. The DDM sees a dataset as collection of files movable across the GRID. An ATLAS physics dataset is a DDM dataset plus associated meta-information.

Datasets are made of datablocks, where a datablock is a group of files that cannot be moved independently, and their content does not change with time. In other terms a datablock is the smallest storage entity used by ATLAS. Datasets are made out of an integer number of datablocks. According to the Luminosity Task Force report, datablocks shall contain an integer number of luminosity blocks. Therefore, datasets will contain an integer number of luminosity blocks. Thus it is possible to calculate the integrated luminosity for a dataset. An example of this is given in the use-cases provided in section 9.

Rules need to be in place to assemble datablocks into datasets. For instance datablocks need to be of a compatible event type (eg do not mix ESD and AOD datablocks). The metadata information for a dataset is defined here in an implementation neutral way and the database services operating at the offline level should provide and utilize this information. Table 8 summarizes the metadata information saved by the offline. This is a minimal set that is based on the experience gained using detector simulation in distributed production during the past few years. We expect this information to grow as we initially gain experience with the offline software during the Computing System Commissioning at first and later with the real data environment.

8.2. Dataset Identification and Naming

Datasets are identified by a Unique Identifier (UID) and by a name that is given for convenience, as it may be easier for a human to identify a dataset using a string rather than an identifier. There should be a one-to-one correspondence between UID and name. The name string normally contains information that is stored in the dataset metadata, to allow human users to obtain minimal information without having to access the metadata database (an operation that may require browsing web pages). This string should be formatted according to specific rules that depend on the project being considered (eg CSC, cosmic data, ...) as described in section 8.5. As an example for the MC exercise carried on during 2006 with software release 11 a typical dataset name was:

csc11.005009.J0_pythia_jetjet.digit.RDO.v1100420

and composed of five "." separated fields, in the following order:

- dataset short Project name (eg "csc11" in the example)
- dataset number (eg "005009" in the example)
- dataset number Physics description (eg "J0_pythia_jetjet")
- dataset file types ("digit.RDO" in the example)
- Release used to produce this dataset

Issue for follow up: In order to limit the length of dataset names and agree on conventions a document on logical file names and physical file names is in preparation[1].

8.3. Dataset Composition, Content and Description

A dataset is composed of datablocks, the list of datablocks should be provided as part of the metadata. Since datablocks are normally created by a single production system task, this can also be used to navigate this information into the production database. Although this is a frequently used feature, we regard it as an implementation detail, that goes beyond the scope of this task force.

The total number of files in a dataset, the number of events, and possibly the total size in Gbytes should be provided. Additional information that can be calculated from the datablock content, such as list of run numbers, or event number range, can be provided to facilitate searches in the metadata database, AMI.

Since a dataset has to be composed of files that are output of the same offline processing stage, the type of data composing the dataset is also saved (eg, RawData, ESD, AOD, ...).

Finally the description of the dataset content is given a string field. This field may contain links to a web (or wiki) page, however we recommend that minimal information is provided also in the metadata itself, due to the natural volatility of web information. A mutable history field should also be provided for information to be added at a later stage to act as sort of dynamic logbook for the dataset.

8.4. Dataset Management Information

Metadata information need to be saved to identify the creation and subsequent operations made on datasets as well as to identify responsible persons or groups for a dataset. The name of the DDM tool (currently DQ2) used to create a dataset should be recorded in the metadata, as this tool may change with time, and the technicalities connected with the retrieval of a dataset content may depend on it.

Ownership of datasets should be established at the group level, where a group can be either a physics group, a subgroup thereof, or some other group of users. Each group should have a manager that is responsible for their datasets at a given point in time. Managers may change with time, while groups should have a longer lifetime. The user creating a dataset can be recorded as this is a single point information.

During different processing stages a dataset can be still accepting the insertion of new datablocks, or may be closed or frozen. The difference between closed and frozen is that a closed dataset can be re-opened by a group or ATLAS production manager, while a frozen dataset is immutable. Time stamps corresponding to operations to a dataset (creation, last update, closing, freezing) should be recorded

A quality flag of the dataset should be used to monitor the relevance of the dataset itself. Datasets can become obsolete (as a old version of the offline software is used to process them), or can be marked as “invalid”, as a wrong set of configuration parameters was used. In the latter case datasets can be masked away by the metadata user interface, or may be marked for deletion by the DDM system.

8.5. Project Information

Information on how a dataset is created should be specific to a particular project. So far we have experience in handling datasets created by offline processing of MC simulated events and of combined test beam. However we expect this type of information to evolve to be tailored to the specific needs while we move on to data taking.

[1] Alexei Klimentov, private communication.

Each project should have a name that is a short string to characterize the project itself. So far we used strings like "csc11" or "rome" to identify MC simulation projects, or "ctb" for combined test beam, and so on. Normally datasets have numbers that identify them within a project, so far we used 6 digit numbers and we attempted to avoid overlap of numbers between different projects for practical purposes. Also project production managers have been dividing the dataset numbers into ranges, to assign different ranges to different subprojects. Datasets have a short description, in the form of a non space-separated string to be used as a part of the dataset name itself for quick identification by the human users. Relevant to the project information is also the identification of the software used to create a dataset. At present this includes, the software release, the database release, the transformation script used and the version of the geometry; we expect the version of the condition database to be included soon.

Since projects, their definition, and the way datasets are created will change with time, this part of the metadata information should be kept flexible. Other metadata information is created during offline processing to record the values of the configurable parameters of the production scripts, like for instance the setting of reconstruction parameters. These configuration parameters may change with the software releases, so this metadata information should be kept flexible to accommodate new or modified information.

Table 8 Metadata information saved by the offline.

Name	Description	Origin	Destination
UID	Dataset unique identifier	Production DB	Metadata DB
Name	dataset name according to convention	Production DB	Metadata DB
Datablock list	Datablock composition of dataset	DDM	Metadata DB
totalFiles	number of files	DDM	Metadata DB
totalEvents	number of events	DDM	Metadata DB
dataSize	Size of dataset	DDM	Metadata DB
runList	list of run numbers	Conditions DB	Metadata DB
dataType	Data format (eg RDO, ESD, AOD, …)	Production DB	Metadata DB
Description	Dataset description (and a pointer to it eg www)	User input	Metadata DB
History	Free text	User input	Medata DB
DDMInstance	Name of DDM tool	Production DB	Metadata DB
Owner	Dataset group owner	Production DB	Metadata DB
UserCreation	Name of user creating dataset	Production DB	Metadata DB
Usergroup	Access list for the dataset	Dataset manager	Metadata DB DDM
Dataset Status	Open, closed, locked, …	Dataset Manager	Metadata DB DDM
Creation Time	Record creation time	DDM	Metadata DB
Update time	Record update time	DDM	Metadata DB
Freeze Time	Record freezing time	DDM	Metadata DB

Quality of dataset	Quality of dataset (good, obsolete, invalid, ...)	Dataset manager	Metadata DB
ProjectName	Project description (csc, susyfilters, …)	Production DB	Metadata DB
Project Number	Dataset number within a project	Production DB	Metadata DB
Short description	Short description of dataset	Production DB	Metadata DB
Software release	Software release used for production	Production DB	Metadata DB
Database release	Database release used for production	Production DB	Metadata DB
Production script	Script used for producing the datasets	Production DB	Metadata DB
Geometry	Geometry tag to be used for the dataset	Production DB	Metadata DB
Conditions DB Version	Version of the conditions DB used	Production DB	Metadata DB

9. Analysis Use Cases

During data analysis, the main task of the Metadata is to provide a list of datasets which fulfils certain conditions (sub-detector status,...) and the associated luminosity. Two main use cases are presented in this document.

9.1. Calibration and Cross-section Measurement of Inclusive Z-> ee Production

The electron selection needs the information from the inner tracker, the electromagnetic calorimeter and at least the first sampling of the hadronic calorimeter but one could survive without the muon detector. The metadata should provide a list of datasets, which fulfil these conditions (use metadata described in 5.3). A luminosity block will contain few di-electron events (1 Hz at 10^{33} cm^{-2} s^{-1}). The extraction of the cross-section needs the luminosity information described in section 5.

The calibration of the electromagnetic calorimeter will need only information the status of this detector. More refined information will be needed, e.g. High-Voltage values and temperature of the Liquid Argon (in case they change). The metadata system proposed allows retrieving the detailed detector status information (see 5.3). The access to calibration runs is needed to understand possible variations of the calibration constants.

At the end of the analysis, results about the electromagnetic calorimeter quality should be added to the metadata information (as pointed out in 5.5).

9.2. A Data File is Inaccessible or Unreadable

There are a number of scenarios in which an analysis task may terminate without having processed all of its intended input: one or more sub-jobs may not run or may terminate prematurely, one or more files may be unreachable, or a particular file may be unreadable. It is the responsibility of the ATLAS distributed production system to report failures in such a way that retry and recovery are possible. A task that involves processing a dataset of 100 files may be decomposed into 100 jobs, one per input file. The production system should retry until the task is complete and all 100 files have been processed, but if the task is interrupted or cancelled (perhaps by the user), it must be straightforward to determine which files have not been processed, and to define a new task to process those files and no others. Because there will be many replicas of AOD and DPD files and at least two replicas of ESD files, circumstances in which a persistent user cannot eventually process all intended input data should be rare.

If, for whatever reason, an analysis processes an incomplete production dataset as input, the metadata system proposed in this document contains sufficient file-level metadata to allow cross-section calculation. For any production file, file-level metadata (see Section 4.2) includes the run(s), luminosity block ranges, and trigger chains used to produce the file, allowing the physicist to adjust the denominator of the cross section to account for data from any missing or otherwise omitted file. (If data from a given luminosity block are spread across multiple files, some of which have been successfully processed and some of which have not, it may be necessary to remove successfully-processed data from the cross section calculation as well, but the same metadata make this adjustment possible.)

A missing RAW file from one of N SFOs is, from the point of view of the metadata system, no different than any other circumstance in which a file containing some, but not all, of the events from a given range of luminosity blocks and triggers may be inaccessible. The proposed metadata system retains sufficient file-level information about run, luminosity block, and trigger content to allow one to identify which other files to omit from one's sample, or to adjust cross-section calculations to compensate, and the event index (Section 4.1) contains more specific information about triggers satisfied by individual events within the missing file. While this scenario raises potentially important operational, policy, and bookkeeping questions, the metadata requirements are the same as for other missing file use cases.

10. Acknowledgements

The authors would like to thank Solveig Albrand and Beate Heinemann for their help in gathering the information presented in section 8 and appendix A.

11. References

[1] ATLAS Raw Event Format in Trigger & DAQ, https://edms.cern.ch/cedar/plsql/doc.info?cookie=6027992&document_id=455903&version=6

[2] Report of the Event Tag Review and Recommendation Group, https://uimon.cern.ch/twiki/bin/viewfile/Atlas/FeedBackForTags?rev=1.2;filename=TagFinalReport-v1.6Final.pdf

[3] Luminosity Task Force report, http://indico.cern.ch/getFile.py/access?contribId=s0t2&sessionId=s0&resId=0&materialId=0&confId=a062711

[4] Data Streaming study group report, in preparation

[5] Run Structure working group report, https://edms.cern.ch/cedar/plsql/doc.info?cookie=5503863&document_id=749421&version=2

[6] Atlas Dataset Definition, SWIG, Note 2005-10-29, https://uimon.cern.ch/twiki/bin/viewfile/Atlas/SoftwareIntegration?rev=1.1;filename=SWING_Oct_31_2005.pdf

[7] H. P. Beck, S. Gadomski, “Format of the raw event data files written by ATLAS TDAQ applications”, Data Collection note 066, EDMS ID 580290, https://edms.cern.ch/document/580290

Appendix A) From Raw Data to Reconstruction in CDF and D0

CDF

1/3 of the data undergoes a special offline processing to provide calibration data. This happens every ~6 weeks.

The off-line processing is re-run using new calibrations, then validated.

There is a special group providing a “good run list” in consultation with detector experts. This list becomes available typically 6-8 weeks after data taking, and ~2 weeks after final processed data are provided.

The Physics Coordinator runs a “validation meeting” whenever a significant chunk of new data becomes analyzable. This open meeting is used to “sign off” on that chunk of data.

D0

Does not allow people to use any data until “final calibration” is in place. The final processing happens typically ~ 6months after data taking.

Calibrations are derived from physics samples in several channels, on a time scale of 6months. The good-run list becomes available at the end of this period. Then the final processing is done.

Earlier data processing is used only for quality studies, never for physics or publications.