

The raw event format in the ATLAS Trigger & DAQ

Authors: C. Bee, D. Francis, L. Mapelli, R. McLaren, G. Mornacchi, J. Petersen, F. Wickens

Abstract

This note presents the ATLAS raw event format. It covers the format of data from the ReadOut Drivers to the output of the Event Filter. It does not cover the detector specific event data.

EDMS: ATL-D-ES-0019
ATLAS Communication: ATL-DAQ-98-129
Major version: 2.4
Date: 2004-02-01

Summary document Change Record

Table 1. Summary document Change Record.

1. Document Title: The raw event format in the ATLAS Trigger & DAQ			
2. Document Reference Number		ATL-D-ES-0018	
4. Issue	5. Revision	6. Date	7. Reason for change
1	0	01 Apr '97	Birth.
1	1	07 Oct. '97	General update of all sections.
1	2	20 Oct. '97	Muon sub-detector IDs changed at the request of S. Falciano.
1	3	14-Aug. '98	Add an offset to ROD trailer that indicates the relative order of Data/status information. General clean-up. Appendix with an initial header file and an appendix of an example use of the header file.
1	4	05-Sept. '98	Redefined last word of ROD trailer. Comments on sub-detector ID's from Philippe Farthouat. Comments from Jorgen Petersen.
1	5	15-Oct. '98	Tidy-up ready for release as ATLAS note Remove Appendices.
2	0	11-Mar. '02	<p>Include feedback on version 1.5 from detector community</p> <p>Increase scope to include Level 2</p> <p>Change from DAQ -1 specific terminology</p> <p>Re-define Source ID</p> <p>Change Level 1 ID element to be defined as the combination of the 24-bit L1ID and the 8-bit ECRID.</p> <p>Add mechanism to determine byte ordering dynamically</p> <p>Remove section on ROL implementation</p> <p>Change unit of Total fragment size and Header size element to be 32-bit integer (see section section).</p> <p>Re-define the Format version number so that it may also be used to identify the format version of the detector Data.</p> <p>Distribute to author list</p> <p>General Distribution</p>
2	2	11-Oct. '02	<p>Added Appendix A on ROL implementation issues.</p> <p>Clean-up of section 2.1 (main requirements).</p> <p>Implementation of Source ID element re-defined, i.e. Module ID now byte wide, see section 5.2.</p> <p>Global event ID removed from ROS specific header, section 5.10.3.</p> <p>Introductory text in section 5 re-written.</p>

Table 1. Summary document Change Record.

2	4	1 Feb. '04	<p><i>Description of "Format version number" element re-written, section 5.6.</i></p> <p><i>Corrected description of Bunch Crossing ID, Tables 8, 9 & 10.</i></p> <p><i>Table 3 defining values for Sub-detector IDs updated to match known TTC partitions.</i></p> <p><i>Error in description of extended level 1 ID corrected, sections 4 & 5.10.1.</i></p> <p><i>Clarified meaning of Detector Event Type element, section 4.</i></p> <p><i>Remove Level 1 Trigger Info. from Full Event Specific header, section 5.10.1.</i></p> <p><i>Initial values and meanings for generic status field (adopted from Level1 - DataFlow interface document), section 5.8.</i></p> <p><i>Remove LVL2-Data and LVL2-Result, table 2.</i></p> <p><i>Added RoI Builder Module Type, table 4.</i></p> <p><i>Deleted sections 5.10.6 and 5.10.7.</i></p> <p><i>Expanded scope to include output of Event Filter, section 1.3.</i></p> <p><i>Added section on Event Filter Output, section 5.11.</i></p> <p><i>Date and time element in Full Event Specific element (Section 5.10.1) redefined to be the number of seconds elapsed since 00h00.00 on 1st Jan. 1970, i.e. in line with Posix.</i></p> <p><i>Run number added to Generic fragment and ROD fragment (section 3,4 and 5). Remove run number from Full Event Specific element (section 5.11.1) and ROS Specific Header (section 5.11.3)</i></p> <p><i>A default sub-detector type added to Table 3. to be used for equipment which is not specific to any single detector.</i></p> <p><i>Module type 'Level 2 Processor' in Table 4. changed to 'HLT Processor'.</i></p> <p><i>Added 'Event Filter Info' in section 5.11.1 and Table 7.</i></p> <p><i>Changed 'Level 1 ID' to 'Extended Level 1 ID' where appropriate.</i></p> <p><i>Updated some of the references.</i></p> <p><i>Cleaned up some typing errors.</i></p>
---	---	------------	---

1 Introduction

1.1 Purpose of the document

This document describes the event format and its initial implementation in the ATLAS Trigger and DAQ. The proposed format is a further step in defining the final ATLAS raw event format.

1.2 Overview of document

In section 2 the requirements, function, purpose and a high-level description of the event format is given. In section 3 a detailed description is given. Section 4 presents a description of the format of a fragment received by the ROB from the ROD over a ROL and is aimed principally at ROD designers. In section 5 an initial implementation of the event format described in the sections 3 and 4 is given. Appendix A: covers the framing information and transmission errors on the ROL.

1.3 Boundaries

This document relates to the format of data into and out of the: Read-Out Sub-system (ROS), DataCollection sub-system, the LVL2 Selection and Event Filter sub-systems of the Higher Level Trigger (HLT). The framing information, necessary to ensure the correct transmission of data between applications, *e.g.* ROD-to-ROB, is technology specific and therefore not part of the event format.

1.4 Definitions, acronyms and abbreviations

See reference [1].

2 General description

2.1 Requirements

This sub-section lists a set of requirements on the various components of the event format. The categories of requirements follow the guidelines given in [2]. Requirements containing the word *shall* are mandatory. Those containing the word *should* are strongly recommended, justification is needed if they are not followed. Sentences containing the word *may* are guidelines, no justification is required if they are not followed.

The event format shall fulfil the following requirements:

1. The event format *shall* allow the size of an event to increase or decrease depending on the specific data taking configuration.
2. There *shall* be no minimum or maximum event data size implied by the format.
3. The event format *should* provide information redundancy to allow self consistency checks of the event to be made. The consistency checks will be defined at a latter stage in time.
4. The event formatting information *shall* not exceed 20% of the typical full ATLAS event data size.

5. The event format *should* be modular¹.
6. The basic unit *should* be a fragment. Fragments are: data coming from a ROD, ROB or ROS, all the data associated to a single sub-detector and the data input to the Event Filter.
7. The fragments *should* have identical structure.
8. The event format *shall* facilitate the identification of fragments.
9. The event format *shall* provide an event header.
10. The event format *shall* provide the event identifier and trigger type within the event header.
11. The event format *shall* provide a means of identifying whether the event has been corrupted during transmission within the DataFlow, *e.g.* DMA time-out, truncation etc.
12. The event format *shall* provide a means of identifying whether the event has been corrupted due to hardware problems, *e.g.* a bit error.

2.2 Function and purpose

The event format defines the structure of the data at various stages within the Trigger and DAQ and allows elements of the DataFlow and processing tasks to access the data without resorting to the use of other resources, *e.g.* databases. In addition, it defines additional data that is added to the detector data, by elements of the TDAQ, allowing processing tasks to quickly identify the type and origin of event.

2.3 General format

The general format of a Full Event is shown in Figure 1. As can be seen it is built from fragments (see requirement 6. in section 2.1). A Full event is an aggregation of sub-detector fragments and each sub-detector fragment is an aggregation of ROS fragments. In turn, each ROS fragment is an aggregation of ROB fragments. Each of the latter map on to one or more ROD fragments. Note: depending on the architecture of the DataFlow system, a sub-detector fragment may be an aggregation of ROB fragments instead of ROS fragments (not shown in). Each fragment, except the ROD fragment, has a header which contains all the event formatting information. For ROD fragments, hardware considerations have led to the combination of a header and a trailer, however, the general principles are similar and it is the combination of the header and trailer which provide the event formatting information. Details of ROD fragments are given in section 4.

The class diagram of the event format is shown in Figure 2. Referring to the latter, it can be seen that a Full Event is one or more Fragments. A Fragment may be associated to zero or more other Fragments. Each Fragment consists of an instance of the Header class and zero or more instances of the Data class. A Header is an aggregation of Generic and Specific classes. The latter being a generalization of: Event, Sub-detector, ROS, ROB and ROD. Headers are invariant of sub-detectors. However the details of the header, for example in a ROB fragment, may vary with respect to a header in a sub-detector fragment.

1. Adjective, (1) of, relating to, or based on a module or a modulus, (2) constructed with standardised units or dimensions for flexibility and variety in use.

As can be seen from Figure 2 the proposed event format is modular and based on event fragments (see section 2.1). All event fragments have the same structure, except the ROD fragment due to identified implementation issues. This fulfils requirement 7. (see section 2.1).

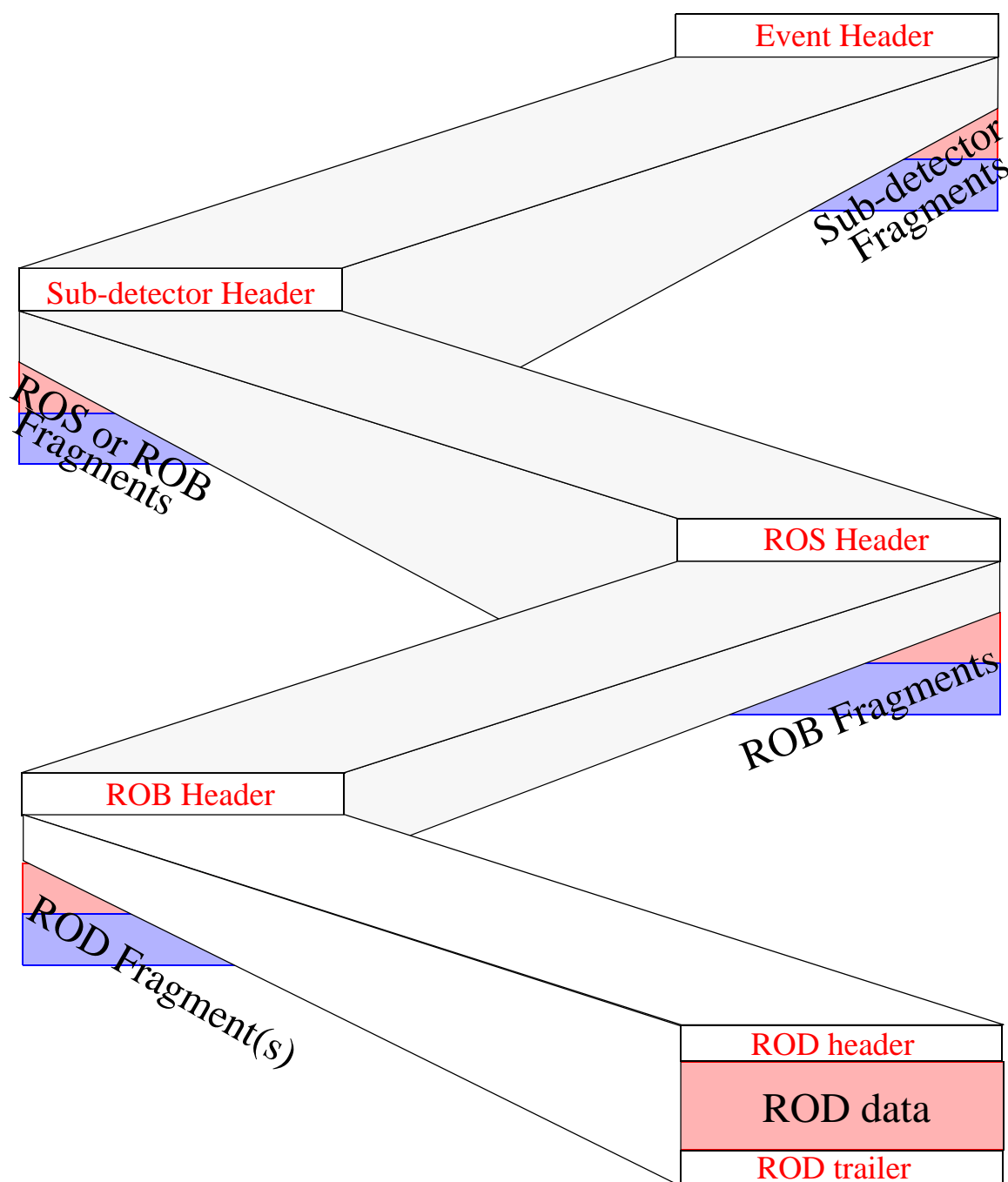


Figure 1. The general event format.

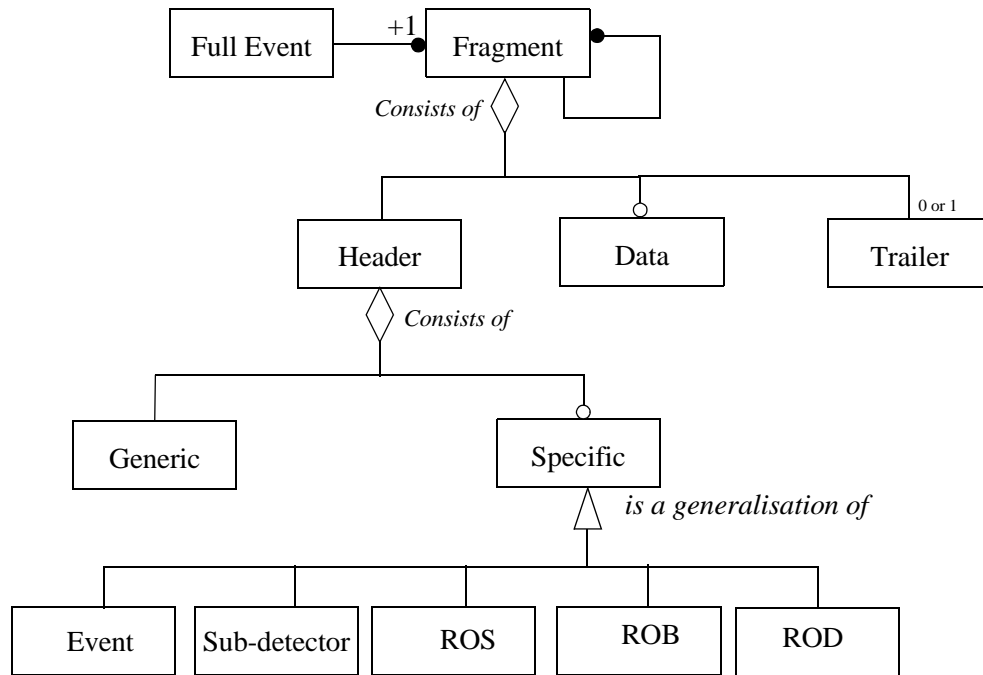


Figure 2. The class diagram of the event format.

3 Header formats

3.1 The Header

The class **Header** is an aggregation of **Generic** and **Specific** parts, see Figure 3. The **Generic** part is the same for all event fragments, except the **ROD** fragment (see section 4). While the **Specific** part allows fragment specific information to be included in the header, e.g. information generated specific to a sub-detector.

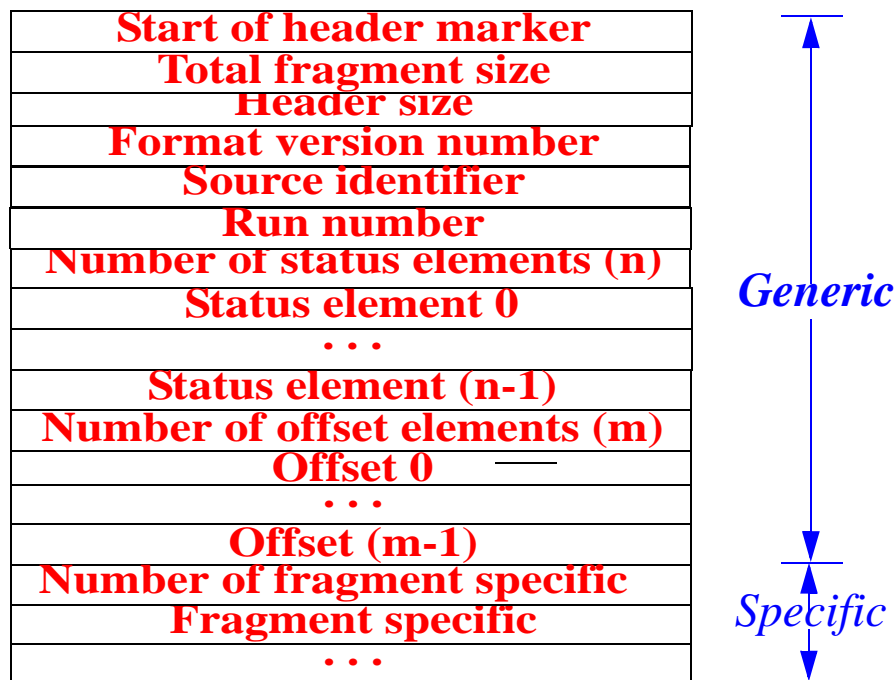


Figure 3. The fragment header.

3.1.1 The Generic component

The Generic component consists of the following elements:

1. *Start of header marker*. This marker indicates the start of a fragment header and is itself part of the header. Hence, it is the first word of a fragment. The value of this element will be unique for each type of fragment, but the structure shall be identical. The structure will allow the endianness of the fragment header to be determined.
2. *Total fragment size*. This element indicates the total size of the fragment, including the Header.
3. *Header size*. The element indicates the total size of the Header.
4. *Format version number*. This element gives the format version of the fragment. For example, if this is a ROB fragment, it defines the format version of the ROB fragment. It allows the format of a ROB fragment to change independently of, for example, a sub-detector fragment.
5. *Source identifier*. This element identifies the origin of the fragment. It consists of a sub-detector ID, Module Type and Module ID. The combination of these fields should allow the Source identifier to be unique across the whole of Atlas. The Module type and ID refer to the module which builds and adds the header to the event fragment.
6. *Run Number*: A 32-bit integer whose value is unique during the lifetime of the experiment.
7. *Number of status elements*. The value of this element is the number of status elements in the Header.
8. *Status element*. This element contains information about the status of the data within the fragment. The structure of this element is specific to the module which builds the header.
9. *Number of offset elements*. The value of this element indicates how many Offset elements are contained within the Header.
10. *Offset*. This element contains an identifier and offset to a fragment contained within this fragment. The offset is relative to the start of this fragment. The use of this element is shown in Figure 4.

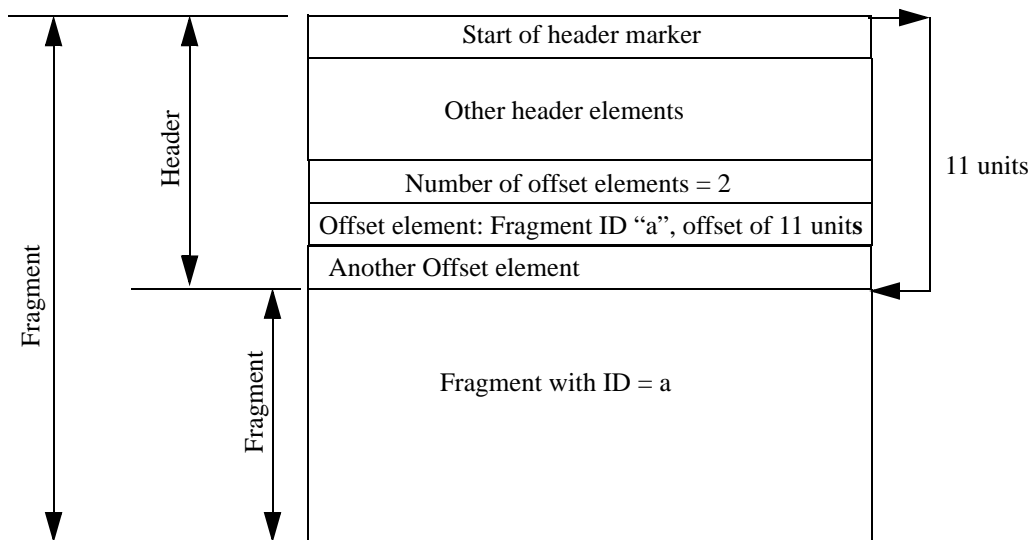


Figure 4. The use of the Offset element.

3.1.2 .The Specific component

Following the Generic component of the header there is a fragment Specific component consisting of:

1. *Number of fragment specific*. An element which gives the number of elements following this element
2. Zero or more elements, as specified in the previous element, which are specific to the type of fragment, e.g. ROB fragment, see section 5.11 for details.

4 ROD data format

The definition of the format of the data transferred between the ROD and ROB must take into account factors such as: the data is formatted in hardware and not necessarily by programmable devices; the information within the header may influence component cost and ROD performance; the actual current understanding of ROD designs (a ROD may not buffer data).

Current designs of the ROD indicate that the data transferred from a ROD to a ROB should have both a Header and a Trailer as shown in Figure 5.

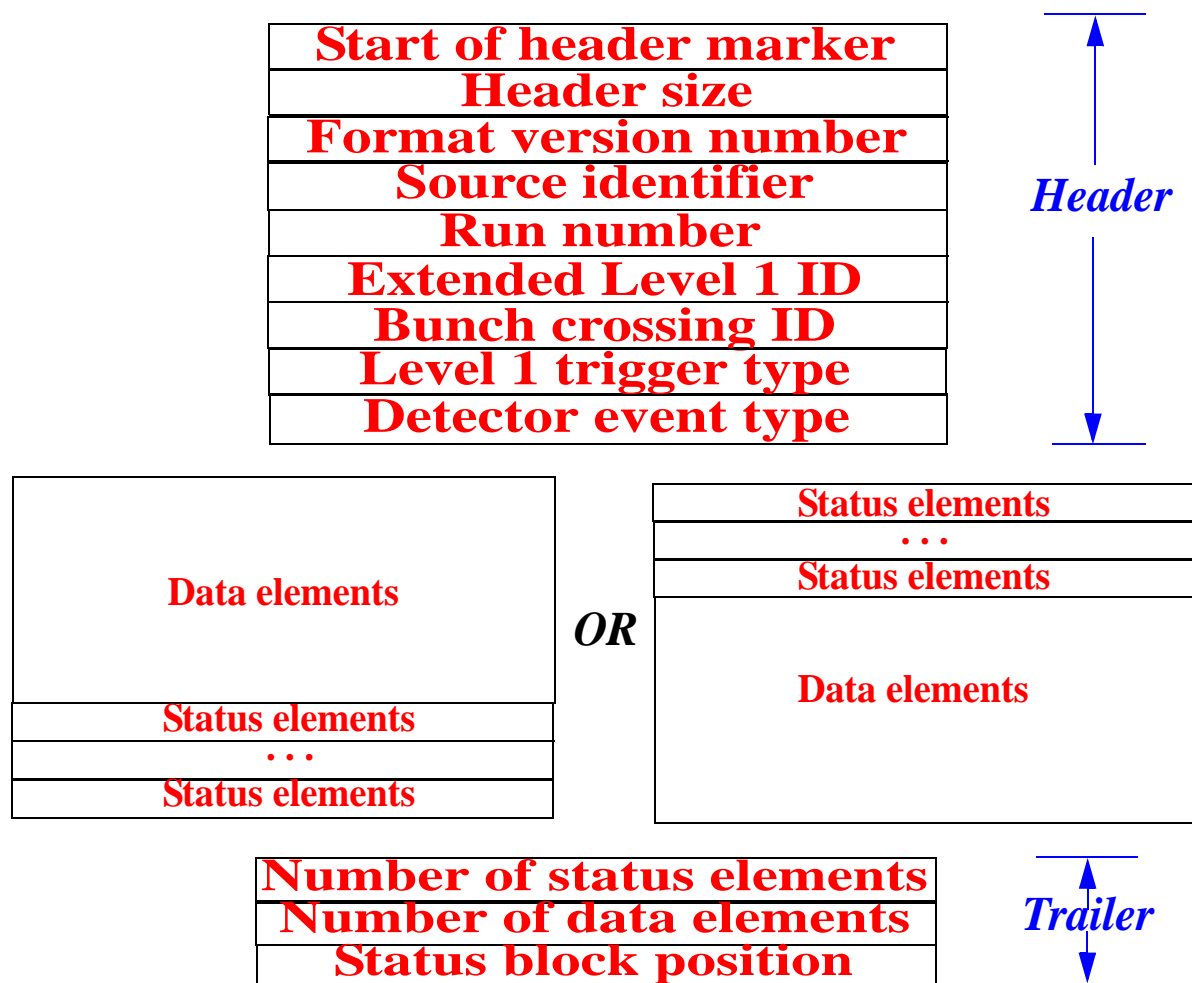


Figure 5. The ROD data format.

The Trailer contains the Number of data elements, Number of Status elements and the status block position. Some detector groups have voiced a preference for having the Status elements proceeding the Data elements. Instead of imposing an order, an additional element, *Status block position*, has been added to the trailer. The value of this element defines the relative order of the Data and Status elements. A value of zero indicates that the status block precedes the data block and a value of one indicates that the status block follows the data block. These two cases are shown in Figure 5 for reasons of clarity. The Data and Status elements are 32-bit integers.

The header is derived from that presented in section 3.1 and the elements have the same meaning. Note, in addition, the value of the Start of Header Marker also identifies the byte order of the ROD fragment Data and Status elements. Within the header four additional elements are explicitly defined, these are:

1. *Extended Level 1 ID*: The Extended L1ID [3] formed by the 24-bit L1ID generated in the TTCrx and the 8-bit ECRID implemented in the ROD.
2. *Bunch Crossing ID*: The 12-bit bunch crossing identifier generated in the TTCrx.
3. *Level 1 Trigger Type*: The 8-bit word generated by the Central Trigger Processor and transmitted by the TTC system [4].
4. *Detector event type*: This element allows additional information to be supplied on the type of event, particularly in the case of calibration events. It allows the detectors to specify the exact type of calibration event that they have generated.

In addition, the first status word must indicate the global status of the fragment, i.e. there shall be at least one status element in a fragment. A non-zero value of this element indicates that the fragment is corrupted, *e.g.* missing data and or bit errors. The exact details of this element are still to be defined.

5 Initial implementation

This section presents an implementation of the event format described in the previous sections. It defines the Start of Header Markers, the Fragment IDs, the sub-detector IDs and the elements specific to the different types of fragments. This implementation is for 32-bit machines and demands that the Generic Header, ROD Header and Trailer are aligned on four byte boundaries. All header and trailer elements are 32-bit integers. For the presentation of the implementation of the event format, Big-endian ordering has been chosen, as it is used in established network header formats.

In this implementation: the ROD, ROB and ROS header are built by the ROD, ROB and ROS respectively; the Sub-detector and Event Header are built by the SFI. It is not excluded that some information from the ROD header and trailer be copied into the ROB header and the ROD header and trailer be discarded. In addition, this implementation does not impose a specific order of the fragments. For example, the first sub-detector fragment in event N may be that of the Pixel Barrel and in event N+1, the first sub-detector fragment may be that of LArg. Barrel right.

The following points have also been taken into account:

- *Bit fields* allow efficient use of memory and matching to hardware-defined data structures. However, they are not portable and therefore not used.
- *Floating point types* are not used in this implementation as they are not portable.
- *Byte ordering*. The endianness of the ROD fragment is Little-endian. This standardisation is defined in [5].

The endianness of the fragments exchanged between the different components of the TDAQ cannot be defined as it cannot be excluded that processors implementing Big or Little-endianness will be deployed. In the absence of a standardisation, the identification of the endian order of a fragment is addressed by the implementation of the Start of header marker, see section 5.12.

- *Alignment*. The implementation demands that all headers are aligned on 4-byte boundaries.

5.1 Start of Header Markers

Each fragment header begins with a Start of Header Marker. These markers fulfil requirements 7, 8 and 9 as described in section 2.1. The markers at each level of the event format are given in Table 2.

The asymmetry in the value of the header marker allows for the byte ordering used in the fragment Header to be identified. Note for the ROD fragment it refers to the byte order of the ROD fragment as a whole.

Table 2. Start of Header Markers.

Fragment Type	Header Marker
<i>ROD</i>	0xee1234ee
<i>ROB</i>	0xdd1234dd
<i>ROS</i>	0xcc1234cc
<i>Sub-Detector</i>	0xbb1234bb
<i>Full Event</i>	0xaa1234aa
<i>LVL1^a Result</i>	0x99123499

a. This is the data sent by LVL1 to the RoI Builder.

5.2 Source IDs

The structure of the Source ID, as shown below, consists of four byte fields. The combination

Byte	3	2	1	0
	Reserved	Module Type	Sub-detector ID	Module ID

of these four fields allows the Source ID to be unique across all sub-detectors. The possible values of the Sub-detector ID and Module Type are defined in section 5.3 and section 5.4. The values that may be assigned to the Module ID are free to be defined by the system or sub-system implementers concerned. The third byte is reserved and should be initialised to a value of zero.

5.3 Sub-Detector IDs

A proposal for sub-detector IDs is given. The detectors (TRT, Pixel, etc.) and TDAQ assigned a unique major-id and up to fifteen possible minor-ids. These values are shown in Table 3.

Table 3. Sub-detector IDs.

Detector		ID
<i>Full Event</i>		<i>0x00</i>
<i>Pixel</i>	<i>Barrel</i>	<i>0x11</i>
	<i>Forward A side</i>	<i>0x12</i>
	<i>Forward C side</i>	<i>0x13</i>
	<i>B-layer</i>	<i>0x14</i>
<i>SCT</i>	<i>Barrel A side</i>	<i>0x21</i>
	<i>Barrel C side</i>	<i>0x22</i>
	<i>Endcap A side</i>	<i>0x23</i>
	<i>Endcap C side</i>	<i>0x24</i>
<i>TRT</i>	<i>Barrel A side</i>	<i>0x31</i>
	<i>Barrel C side</i>	<i>0x32</i>
	<i>Endcap A side</i>	<i>0x33</i>
	<i>Endcap C side</i>	<i>0x34</i>
<i>LAr</i>	<i>EMB A side</i>	<i>0x41</i>
	<i>EMB C side</i>	<i>0x42</i>
	<i>EMEC A side</i>	<i>0x43</i>
	<i>EMEC C side</i>	<i>0x44</i>
	<i>HEC A side</i>	<i>0x45</i>
	<i>HEC C side</i>	<i>0x46</i>
	<i>FCAL A side</i>	<i>0x47</i>
	<i>FCAL C side</i>	<i>0x48</i>
<i>TileCal</i>	<i>Barrel A side</i>	<i>0x51</i>
	<i>Barrel C side</i>	<i>0x52</i>
	<i>Extended A side</i>	<i>0x53</i>
	<i>Extended C side</i>	<i>0x54</i>

Table 3. Sub-detector IDs.

Detector		ID
<i>Muon</i>	<i>MDT Barrel A side</i>	<i>0x61</i>
	<i>MDT Barrel C side</i>	<i>0x62</i>
	<i>MDT Endcap A side</i>	<i>0x63</i>
	<i>MDT Endcap C side</i>	<i>0x64</i>
	<i>RPC Barrel A side</i>	<i>0x65</i>
	<i>RPC Barrel C side</i>	<i>0x66</i>
	<i>TGC Endcap A side</i>	<i>0x67</i>
	<i>TGC Endcap C side</i>	<i>0x68</i>
	<i>CSC Endcap A side</i>	<i>0x69</i>
	<i>CSC Endcap C side</i>	<i>0x6a</i>
<i>T/DAQ</i>	<i>Calorimeter preprocessor</i>	<i>0x71</i>
	<i>Calorimeter Cluster processor</i>	<i>0x72</i>
	<i>Calorimeter Jet/Energy processor</i>	<i>0x73</i>
	<i>CTP</i>	<i>0x74</i>
	<i>Muon Interface</i>	<i>0x75</i>
	<i>DataFlow</i>	<i>0x76</i>
	<i>LVL2</i>	<i>0x77</i>
	<i>Event Filter</i>	<i>0x78</i>
<i>Other^a</i>		<i>0x81</i>

a. An example is the beam crate used at test beams.

5.4 Module Type

For the format described here the Module Type is byte wide, allowing a maximum of 256 possible modules to be enumerated. A list of identified module types is enumerated in Table 4.

Table 4. Enumeration of Module Type.

Module Type	Value
<i>ROD</i>	<i>0x00</i>
<i>ROB</i>	<i>0x01</i>
<i>ROS</i>	<i>0x02</i>
<i>RoI Builder</i>	<i>0x03</i>
<i>Supervisor</i>	<i>0x04</i>
<i>HLT Processor</i>	<i>0x05</i>
<i>SFI</i>	<i>0x06</i>

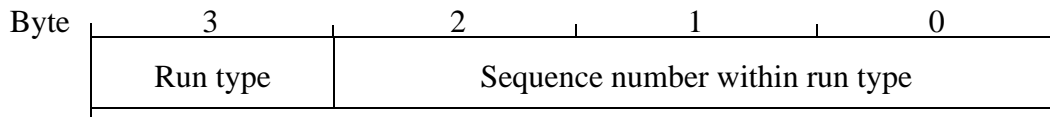
Table 4. Enumeration of Module Type.

Module Type	Value
<i>SFO</i>	<i>0x07</i>
<i>Other^a</i>	<i>0x0a</i>

a. An example of this is a VMEbus module emulating a ROD.

5.5 Run number

The run number is a 32-bit integer. The 8 highest bits are defined by the run control and identify the type of run. Examples of run types: sub-detector calibration; physics running; combined detectors run, e.g. Level 1 calorimeter and a Liquid Argon sub-detector. The value of lower 24-bits represent the ordered sequence of runs within a type. The structure of the run number is shown below.



An enumeration of Run Type is shown in Table 5.

Table 5. Enumeration of Run Type.

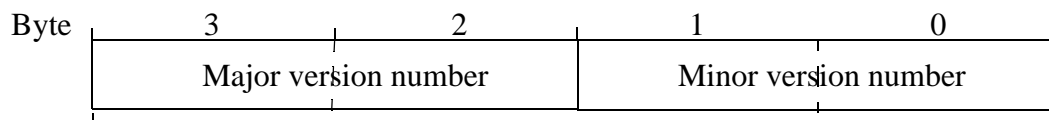
Run Type	Value
<i>Physics</i>	<i>0x00</i>
<i>Calibration</i>	<i>0x01</i>
<i>Cosmics</i>	<i>0x02</i>

5.6 Total fragment and Header size

These elements are each 32-bit integers and their values give the total size of the fragment and the size of the fragment header in units of 32-bit integers.

5.7 Format Version Number

This element consists of two 16-bit fields, as shown below. The combined value of these fields



identifies the fragment format version. The Major version number shall be the same for all fragments in the event, i.e. it refers to the format of the Generic Header, the ROD Header and the ROD trailer. The Minor version number has a value dependent on the fragment type and will be used to identify the format of the specific part of the fragment header and in a ROD fragment the format of the sub-detector Data.

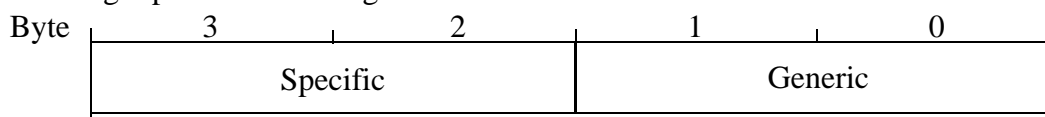
The implementation described in this document defines the Format Version Number to be 2.4.0.0 (0x02040000), i.e. Major version number is 2.4 and the Minor version number is 0.0.

5.8 Number of Status elements

For the initial implementation there should be at least one status element, see below. Therefore this element must have a value greater than or equal to one.

5.9 Status element

This element is a 32-bit integer. There must be at least one Status element in a Fragment. A non-zero value of the first status element indicates that the event fragment is corrupted, *e.g.* truncated. The first Status element, as shown below, is divided into two 2-byte fields labelled Generic and Specific. The values and error conditions indicted by the Generic field are the same for all fragments, while the values and error conditions indicated by the Specific field have meanings specific to the fragment.



The currently defined values and meanings of the Generic field are given in Table 6.

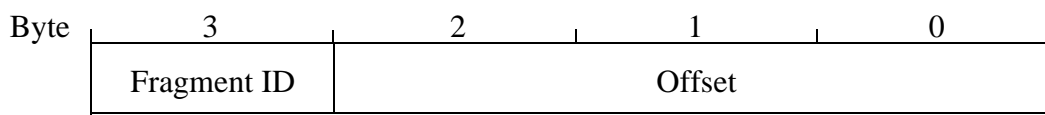
Table 6. Values and meaning for the Generic field of the mandatory first status element.

Generic field value	Description
1	<i>An internal check of the BCID has failed</i>
2	<i>An internal check of the ELIID has failed</i>
4	<i>A time out in one of the modules has occurred. The fragment may be incomplete</i>
8	<i>Data may be incorrect. Further explanation in Specific field</i>
16	<i>An overflow in one of the internal buffers has occurred. The fragment may be incomplete</i>

The structure and values of subsequent Status elements are free to be defined by the designers of the modules which build the specific Event fragment header.

5.10 Offset element

This element is 32-bits wide and has the structure shown below.



Bytes 0 to 2 are the offset to the fragment identified by the value in byte 3 and is in units of 32-bit integers. Byte 3 identifies the fragment found at the offset. For example: if this element is part of a Full Event header then Fragment ID takes the value of a Sub-detector ID; if this element is part of a Sub-detector header then Fragment ID takes the value of a ROS ID; If this element is part of a ROS Fragment header then Fragment ID takes the value of a ROB Module

ID; If this element is part of a ROB Fragment header then Fragment ID takes the value of a ROD Module ID.

5.11 Fragment specific elements

5.11.1 Full Event Specific elements

The Full Event specific elements are defined in Table 7. Each element is a 32-bit integer. The table also presents the required order of the specific elements.

Table 7. Fragment Specific Header for the full event.

Event Header Words	Definition
<i>Date & Time</i>	0xssssssss
<i>Global event ID</i>	32-bit integer
<i>Extended Level 1 ID</i>	0xnnnnnnnn
<i>Level 1 Trigger Type</i>	0x000000tt
<i>Level 2 Trigger Info</i>	0xnnnnnnnn
<i>Event Filter Info</i>	0xnnnnnnnn
	0xnnnnnnnn
	0xnnnnnnnn
	0xnnnnnnnn

- *Date & Time*: This element encodes the date and time as the number of seconds elapsed since 00h00.00 on 1st January 1970. It provides a constantly increasing number with which one may use to time-order events. The Full Event fragment is built by the SFI, hence this element is the time at which the event was built.
- *Global event ID*: The value of this 32-bit integer will be provided by the DFM component of the Event Building subsystem. The value will be unique within a run.
- *Extended Level 1 ID*: The extended LVL1 ID [3] formed by the 24-bit L1ID generated in the TTCrx and the 8-bit ECRID implemented in the ROD.
- *Level 1 Trigger Type*: An 8-bit word as generated by the Central Trigger Processor and transmitted by the TTC system [4]. The remaining 24-bits are un-used.
- *Level 2 Trigger Info*: Summary information regarding the event. The element is one 32-bit integer. The possible values that this element may take are still to be defined.
- *Event Filter Info*: Summary information regarding the event. This element is four 32-bit integers in size. The possible values that this element may take are still to be defined.

5.11.2 Sub-Detector Specific Header

The fragment specific elements for a Sub-detector header are defined in Each element is a 32-bit integer.

Table 8. Fragment Specific Header for a sub-detector.

Sub-Detector specific	Definition
<i>Level 1 Trigger Type</i>	0x000000tt

- *Level 1 Trigger type*: see section 5.11.1.

5.11.3 ROS Specific Header

The elements specific to a ROS fragment are shown below. There are currently two elements defined. The Extended Level 1 ID is defined in section 5.11.1. The Bunch crossing ID is as defined in section 4. Only the lower 12-bits of this element are used, the upper 20-bits are set to zero.

Table 9. Fragment Specific Header for a ROS

ROS specific	Definition
<i>Bunch Crossing ID</i>	0x00000xxx
<i>Extended Level 1 ID</i>	0xn timer timer

5.11.4 ROB Specific Header

The elements specific to a ROB fragment are shown below. There are currently four elements defined. The Extended Level 1 ID, the Bunch crossing ID, the Level 1 Trigger Type and the Detector specific Type, see section 5.11.1, section 5.11.3 and section 4.

Table 10. Fragment Specific Header for a ROB.

ROB specific	Definition
<i>Extended Level 1 ID</i>	0xn timer timer
<i>Bunch Crossing ID</i>	0x00000xxx
<i>Level 1 Trigger Type</i>	0x000000tt
<i>Detector Event type</i>	<i>to be defined</i>

5.11.5 LVL1-Result Specific Header

The elements specific to a LVL1-Result fragment are shown in Table 11. (this is the fragment sent from the LVL2 Supervisor to the LVL2 Processor informing the latter of the LVL1 Regions of interest for the event). There are currently three elements defined. The Extended Level 1 ID, the Bunch crossing ID and the Level 1 Trigger Type, see section 5.11.1, section 5.11.3 and section 4.

Table 11. Fragment Specific Header for the LVL1-Result.

LVL1-Result specific	Definition
<i>Extended Level 1 ID</i>	0xn timer timer
<i>Bunch Crossing ID</i>	0x00000xxx
<i>Level 1 Trigger Type</i>	0x000000tt

5.12 Event Filter Output

The input to the Event Filter is a Full Event fragment. The output of the Event Filter shall be the same Full Event fragment with an additional Sub-detector fragment appended. To mini-

mise manipulation of the Full Event fragment header and to take into account the appending of the additional detector fragment, when the Full Event fragment header is built by the SFI it shall place in the header an Offset element whose Fragment ID has a value equal to that of the “TDAQ Event Filter” Sub-detector ID and an Offset value equal to the value of the second element of the fragment header, i.e. “Total fragment size” element.

5.13 ROD Header and trailer

The initial implementation of the ROD header and trailer has been given in section 4. These elements, including the Data and Status elements, are 32-bit integers, *e.g.* The Level 1 Trigger type is an 8-bit value, therefore the remaining 24-bits are un-used.

Appendix A: Framing

In transmitting an event or an event fragment between: elements of the Trigger and DAQ; the ROD and the ROB; elements within the DataFlow, the fragment must be framed by technology specific information. This framing information is not part of the event format, it is specific to the link technology used and will be removed by a receiving element. Any status information contained in the framing information will, of course, be preserved.

Current ROLs are an implementation of the S-LINK specification. This section spells out the use of S-LINK control words for framing the ROD fragments and also the detection of errors during transmission over an S-LINK ROL.

Each ROD fragment is preceded and terminated by a single S-LINK control word. These control words are referred to as the Beginning of Fragment and the End of Fragment. The words are 32-bits integers and take the values shown in Table 12.

Table 12. Beginning and End of fragment control words.

Control word	Value
<i>Beginning of Fragment</i>	<i>0xb0f0rrrr</i>
<i>End of Fragment</i>	<i>0xe0f0rrrr</i>

The lower sixteen bits of the Beginning of Fragment and End of Fragment control words are used by S-LINK to report transmission errors and are therefore subsequently reserved. All bits should be set to zero prior to transmission of the control words. Only bits [1..0] are currently used. The description and meaning of these bits is shown in Table 13.

Table 13. The meaning of the reserved bits in the S-LINK control words.

	Meaning when	
	Value is 0	Value is 1
<i>Bit 0</i>	<i>Previous data fragment ok</i>	<i>Transmission error in previous data fragment</i>
<i>Bit 1</i>	<i>Control word ok</i>	<i>Transmission error in control word</i>

References

- [1]*ATLAS High-Level Trigger Data Acquisition and Controls Technical Design Report, Appendix B*, CERN/LHCC/2003-022 (2003)
- [2]C. Mazza et. al., *Software Engineering Standards*. Prentice Hall. ISBN 0-13-106568-8
- [3]R. Spiwoks, Presentation given to the Front-end Electronics Co-ordination - 27/02/02
(c.g. <http://documents.cern.ch/AGE/current/fullAgenda.php?ida=a02188&>)
- [4]Definition of the trigger-type word, ATL-DA-ES-0022
- [5]R. McLaren, *ATLAS Read Out Drivers: Endianness*, EDMS Note, ATC-TD-EC-0001, (2003)