ATLAS

ATLAS TDAQ <sub-group>

<component> Requirements

Document Version:	1.0
Document ID:	ATLAS-TDAQ-2002-XXX
Document Date:	Last modification date
Document Status:	Draft | Final

Abstract

Abstract text for the Requirements document.

Institutes and Authors:

Institute 1: A. Author, B. Author, C. Author

Institute 2: A. Author

Institute 3: A. Author, B. .Author

etc...

Table 1 Document Change Record

Title:	ATLAS TDAQ <sub-group> <component> Requirements		
ID:	ATLAS-TDAQ-2002-XXX		
Version	**Issue**	**Date**	**Comment**
1	1		

A different version number should be given to the document if substantive changes to the contents of the document have been made. Different issue numbers within a given version indicate minor changes only such as spelling and grammatical corrections.

Note on red text: The red text provides an explanation of the suggested contents for each section and some examples. In the Framemaker template the red text is defined as "Conditional text" with the "Template Guide" tag. All the Template Guide text may be displayed or hidden by the following menu options:

Special/Conditional Text.../Show | Hide...

The colour of the Template Guide text may be changed in the following way:

Special/Conditional Text.../select "Template Guide"/Edit Condition Tag...

1 Introduction

Brief introductory text about the component.

1.1 Purpose of the document

This document presents the Requirements for the <component> of the ATLAS TDAQ <sub-group>. They shall be the basis for the design and implementation of the <component> in the context of the ATLAS Trigger DAQ system.

Include a very short description of what the component is responsible for (3-5 lines).

1.2 Glossary, acronyms and abbreviations

Provide definitions for all terms, acronyms and abbreviations used in the document. Any existing glossaries which are used should be referenced, to avoid repetition here.

1.2.1 Glossary

For long descriptions use a pair of Dfn1Part1Term/Dfn1Part2Desc paragraph formats:

Example: **Farm**
a set of computing nodes linked by a network or a bus

Something else to be defined
and here is the description for something else

1.2.2 Acronyms and Abbreviations

For short descriptions use a pair of Dfn1Part1TermRIH/Dfn1Part2Desc paragraph formats:

CERN European Laboratory for Particle Physics

ASA Another Silly Acronym

1.3 References

List all external documents referenced in this document.

1

2

3

4

5

2 General Description [optional]

This section is optional. For simple components, all the necessary description could be included in the introduction. However, it should include short statements covering the information outlined in the following sub-sections.

2.1 Context of the <component>

Describe related external systems and subsystems. General description of context in which the component will operate. Description of external systems with which the component is expected to interact. Description of interfaces of external systems with the component.

2.2 General capabilities of <component>

Describe the main capabilities required and why they are needed.

2.3 General constraints on <component>

Describe the main constraints that apply and why they exist.

2.4 General assumptions and dependencies

Describe the assumptions that the component has to make concerning external systems. For example, the services the component requires external systems to implement.

2.5 User characteristics

Describe who will use the component and when.

3 Specific Constraints, Assumptions/Dependencies, Use Cases and Requirements

For a small component, a single level of description may be sufficient in each category. For a complex component, an optional hierarchical listing within each category may be useful. This would start with for example, the constraints relevant to the whole component, followed by a breakdown, in sub-sections of the constraints for the various sub-components.

3.1 Constraints

A constraint is something that affects the way that requirements are met. It imposes restrictions on the design of the system that do not affect the external behaviour of the system, but must be fulfilled to meet technical or project obligations. Typical constraints are time, money, technology, interaction with already existing systems.

Each constraint must have a unique identifier (Item Identifier paragraph tag in Framemaker) to facilitate tracing through subsequent phases of the software development process. A title for the constraint may be added, or left blank. The strictness with which the constraint is to be applied shall be indicated by correct use of the terms "shall", "should", "may", "can" in the constraint statement (see OnlineSW Requirements Checklist for further information: http://atddoc.cern.ch/Atlas/DaqSoft/sde/inspect/Requirements_checklist.html). Essential constraints are non-negotiable; others may be less vitally important and subject to negotiation. Any other relevant information (e.g. related requirements) can be added in a note.

CO001 ***An Example:*** **Time Restriction**

The <component> shall be operational on 1st March 2002.

CO002

An Example: The <component> software should be written in C++.

CO [id] Constraint Title [optional]

[Constraint Statement]

Note Optional extra information...

3.1.1 <sub-component 1> Constraints [optional]

CO101 ***An Example:*** **Time Restriction**

The <sub-component 1> shall be operational on 1st January 2002.

3.1.2 <sub-component 2> Constraints [optional]

CO201

3.2 Assumptions and Dependencies

An assumption/dependency is a requirement or constraint the component puts on an external system with which the component will interact. The external system has to implement these in order for the component to fulfil its role correctly. N.B. It is important that these Assumptions and Dependencies are made known to those responsible for the design and development of the relevant external components.

Each assumption/dependency must have a unique identifier (Item Identifier paragraph tag in Framemaker) to facilitate tracing through subsequent phases of the software development process. A title for the assumption/dependency may be added, or left blank. The strictness with which the assumption/dependency is to be applied shall be indicated by correct use of the terms "shall", "should", "may", "can" in the constraint statement (see OnlineSW Requirements Checklist for further information: http://atddoc.cern.ch/Atlas/DaqSoft/sde/inspect/Requirements_checklist.html). Essential constraints are non-negotiable; others may be less vitally important and subject to negotiation. Any other relevant information (e.g. related requirements) can be added in a note.

AD001

External component shall deliver the run number to <component> at the start of each new run.

Note Optional extra information...

AD [id] Assumption/Dependency Title [optional]

[Assumption / Dependency Statement]

Note Optional extra information...

3.2.1 <sub-component 1> Assumptions and Dependencies [optional]

AD101 Assumption/Dependency [optional]

[Assumption / Dependency Statement]

Note Optional extra information...

3.2.2 <sub-component 2> Assumptions and Dependencies [optional]

AD201

3.3 Use Cases [optional]

Use cases capture the different things an external "user" wishes to do with the component. The user may be a human or external software/hardware. A use case describes a sequence of actions, performed by the component that yields a result of value to the user. The use case describes how users and the system work together to achieve a particular goal. Use cases can be a useful starting point for deriving functional requirements and as illustrations and examples of the components behaviour. The inclusion of the use case section is optional but authors are encouraged to put in use cases if they have them already prepared.

Each use case must have a unique identifier (Item Identifier paragraph tag in Framemaker) to facilitate tracing through subsequent phases of the software design process. A title for the use case may be added, or left blank. A short description of the use case should be given. Any other relevant information (e.g. related requirements) can be added in a note.

UC001 Use Case Title [optional]

[Use Case Description]

Note Optional extra information...

UC [id] ***An Example:*** **Starting the system**

An operator will use the graphical user interface to request that <component> starts all child processes.

Note Optional extra information....

3.3.1 <sub-component 1> Use Cases [optional]

UC101 Use Case Title [optional]

[Use Case Description]

Note Optional extra information...

3.3.2 <sub-component 2> Use Cases [optional]

UC201

3.4 Functional Requirements

A functional requirement is something that the component must do. It captures the intended behaviour of the component. The behaviour may be expressed as services, tasks or functions the <component> is required to perform.

Each requirement must have a unique identifier (Item Identifier paragraph tag in Framemaker) to facilitate tracing through subsequent phases of the software design process. A title for the requirement may be added, or left blank. A statement of the requirement must be given. Essential requirements shall be marked as such by the correct use of the terms "shall", "should", "may", "can" (see OnlineSW Requirements Checklist for further information: http://atddoc.cern.ch/Atlas/DaqSoft/sde/inspect/Requirements_checklist.html). Essential requirements are non-negotiable; others may be less vitally important and subject to negotiation. Each requirement shall include a measure of priority so that the developer can decide the production schedule. Any other relevant information, such as examples to illustrate the purpose of the requirement can be added in a note.

UR001 Requirement Title [optional]

[UR Statement]

Priority [Priority for incremental delivery]
Note Something...

UR002 *An Example:* Starting child processes

The <component> software shall provide a means to start all child processes.

Priority High

3.4.1 <sub-component 1> Functional Requirements [optional]

UR101 Requirement Title [optional]

[UR Statement]

Priority [Priority for incremental delivery]
Note Something...

3.4.2 <sub-component 2> Functional Requirements [optional]

UR201

3.5 Non-Functional Requirements

A non-functional requirement is a property the component must have. They are additional system attributes. Many categories of non-functional requirements exist: performance, interface, operational, resource, verification, acceptance-testing, documentation, security, portability, quality, reliability, maintainability and safety requirements (see Software Engineering Guides, C.Mazza et al, Prentice Hall, p.99 for more information). They specify how well some behavioural aspect of the system should be accomplished. If desired, the Framemaker "Heading" tag can be used to group the requirements into the different categories.

UR0XX Requirement Title [optional]

[UR Statement]

Priority	[Priority for incremental delivery]
Note	Something...

Performance Requirements [optional]

UR [id] ***An Example:*** **Performance**

All child processes shall be started within a time limit of 1 minute.

Priority	High
Note	Related requirement UR002

Interface Requirements [optional]

Operational Requirements [optional]

Resource Requirements [optional]

etc.

3.5.1 <sub-component 1> Non-Functional Requirements [optional]

UR1XX Requirement Title [optional]

[UR Statement]

Priority	[Priority for incremental delivery]
Note	Something...

3.5.2 <sub-component 2> Non-Functional Requirements [optional]

UR2XX

This document has been prepared using the Requirements Document Template provided and approved by the Atlas TDAQ and DCS Connect Forum. For more information, go to http://atlas-connect-forum.web.cern.ch/Atlas-connect-forum/ .
The template is based on the SDLT Single File Template that has been prepared by the IPT Group (Information, Process and Technology), IT Division, CERN (The European Laboratory for Particle Physics). For more information, go to http://framemaker.cern.ch/.