

High level design of non-blocking availability module for Colibri port

By Peter Braam <peter_braam@xyratex.com>

Date: 2012/07/03

Revision: 1.0

[Text in square brackets and with a light-green background is a commentary explaining the structure of a design document. The rest is a fictional design document, used as a running example.]

This document presents a high level design (HLD) of non-blocking availability (NBA) module of Mero core. The main purposes of this document are: (i) to be inspected by Mero architects and peer designers to ascertain that high level design is aligned with Mero architecture and other designs, and contains no defects, (ii) to be a source of material for Active Reviews of Intermediate Design (ARID) and detailed level design (DLD) of the same component, (iii) to serve as a design reference document.

The intended audience of this document consists of Mero customers, architects, designers and developers.

[High level design of non-blocking availability module for Colibri port](#)

[0. Introduction](#)

[1. Definitions](#)

[2. Requirements](#)

[3. Design highlights](#)

[4. Functional specification](#)

[5. Logical specification](#)

[5.1. Conformance](#)

[5.2. Dependencies](#)

[5.3. Security model](#)

[5.4. Refinement](#)

[6. State](#)

[6.1. States, events, transitions](#)

[6.2. State invariants](#)

[6.3. Concurrency control](#)

[7. Use cases](#)

[7.1. Scenarios](#)

[7.2. Failures](#)

[8. Analysis](#)

[8.1. Scalability](#)

[8.2. Other](#)

[8.2. Rationale](#)

[9. Deployment](#)

[9.1. Compatibility](#)

[9.1.1. Network](#)

[9.1.2. Persistent storage](#)

[9.1.3. Core](#)

[9.2. Installation](#)

[10. References](#)

0. Introduction

[This section succinctly introduces the subject matter of the design. 1--2 paragraphs.]

[A note on notation.]

The most unusual feature of the present specification format is a system of tagging used for requirements tracking purposes. Overall goals of requirements tracking are described in [this document](#). The crux of the matter is the idea of using tags to track dependencies between system components and between layers of the system specification (architecture, high level design, detailed level design, implementation). A part of system architecture is a [Summary Requirements table](#) that contains basic architecture level requirements for many system components. Additional requirements for a particular component are collected during requirements analysis phase of development. A requirement is labelled by a tag having form [r.component.tag]. [Requirements](#) section of an HLD collects together all requirements for the component. As design is elaborated in functional and logical specifications, dependencies on properties of other components can arise. Such a dependency is recorded as a [u.component.tag] label in a footnote, added at the point where the dependency is introduced, where "tag" somehow denotes property the design depends on. If the property is required by the Summary Requirements table, "ST" is added to the label. All new dependencies introduced by the specification are enumerated in a [Dependencies](#) sub-section of Logical specification.

In addition to requirements for other components, design can introduce dependencies on the lower specification layers of the same component. For example, a high level design specification is usually correct under certain assumptions that can be validated only in detailed level design. A detailed level design specification is similarly dependent on requirements to component implementation. For simplicity, such intra-component dependencies are not recorded as footnotes. Instead they are enumerated a [Refinement](#) sub-section.

After a design has been described, it is necessary to show that it conforms to the stated requirements. To this end, a Conformance sub-section describes how requirements are discharged using [i.component.tag] labels.

With this notation in place, the following requirements integrity condition can be checked:

for every [r.component.tag] in the system documentation there is a [i.component.tag], somewhere in the documentation. That is, every requirement is implemented.

Additionally, [u.component.tag] tags allow to back-track every requirement to its particular users, providing a way to estimate how the system would be affected if the requirement is to be abandoned or modified.

The following color marking is used in this document: incomplete or todo item, possible design extension or future directions.]

1. Definitions

[Definitions of terms and concepts used by the design go here. The definitions must be as precise as possible. References to the [Mero Glossary](#) are permitted and encouraged. Agreed upon terminology should be incorporated in the glossary.]

2. Requirements

[This section enumerates requirements collected and reviewed at the Requirements Analysis (RA) and Requirements Inspection (RI) phases of development. References to the appropriate RA and RI documents should go here. In addition this section lists architecture level requirements for the component from the Summary requirements table and appropriate architecture documentation.]

3. Design highlights

[This section briefly summarises key design decisions that are important for understanding of the functional and logical specifications and enumerates topics that a reader is advised to pay special attention to.]

4. Functional specification

[This section defines a [functional structure](#) of the designed component: the decomposition showing *what* the component does to address the requirements.]

5. Logical specification

[This section defines a logical structure of the designed component: the decomposition showing *how* the functional specification is met. Subcomponents and diagrams of their interrelations should go in this section.]

5.1. Conformance

[For every requirement in the Requirements section, this sub-section explicitly describes how the requirement is discharged by the design. This section is part of a requirements tracking mechanism,

so it should be formatted in some way suitable for (semi-)automatic processing.]

5.2. Dependencies

[This sub-section enumerates other system and external components the component depends on. For every dependency a type of the dependency (uses, generalizes, *etc.*) must be specified together with the particular properties (requirements, invariants) the design depends upon. This section is part of a requirements tracking mechanism.]

5.3. Security model

[The security model, if any, is described here.]

5.4. Refinement

[This sub-section enumerates design level requirements introduced by the design. These requirements are used as input requirements for the detailed level design of the component. This sub-section is part of a requirements tracking mechanism.]

6. State

[This section describes the additions or modifications to the system state (persistent, volatile) introduced by the component. As much of component behavior from the logical specification should be described as state machines as possible. The following sub-sections are repeated for every state machine.]

6.1. States, events, transitions

[This sub-section enumerates state machine states, input and output events and state transitions incurred by the events with a table or diagram of possible state transitions. [UML state diagrams](#) can be used here.]

6.2. State invariants

[This sub-section describes relations between parts of the state invariant through the state modifications.]

6.3. Concurrency control

[This sub-section describes what forms of concurrent access are possible and what forms on concurrency control (locking, queuing, *etc.*) are used to maintain consistency.]

7. Use cases

[This section describes how the component interacts with rest of the system and with the outside world.]

7.1. Scenarios

[This sub-section enumerates important use cases (to be later used as seed scenarios for ARID) and describes them in terms of logical specification.]

Scenario	[usecase.component.name]
Relevant quality attributes	[e.g., fault tolerance, scalability, usability, re-usability]
Stimulus	[an incoming event that triggers the use case]
Stimulus source	[system or external world entity that caused the stimulus]
Environment	[part of the system involved in the scenario]
Artifact	[change to the system produced by the stimulus]
Response	[how the component responds to the system change]
Response measure	[qualitative and (preferably) quantitative measures of response that must be maintained]
Questions and issues	

[[UML use case diagram](#) can be used to describe a use case.]

7.2. Failures

[This sub-section defines relevant failures and reaction to them. Invariants maintained across the failures must be clearly stated. Reaction to [Byzantine failures](#) (*i.e.*, failures where a compromised component acts to invalidate system integrity) is described here.]

8. Analysis

8.1. Scalability

[This sub-section describes how the component reacts to the variation in input and configuration parameters: number of nodes, threads, requests, locks, utilization of resources (processor cycles, network and storage bandwidth, caches), *etc.* Configuration and work-load parameters affecting component behavior must be specified here.]

8.2. Other

[As applicable, this sub-section analyses other aspects of the design, *e.g.*, recoverability of a distributed state consistency, concurrency control issues.]

8.2. Rationale

[This sub-section describes why particular design was selected; what alternatives (alternative designs and variations of the design) were considered and rejected.]

9. Deployment

9.1. Compatibility

[Backward and forward compatibility issues are discussed here. Changes in system invariants (event ordering, failure modes, *etc.*)]

9.1.1. Network

9.1.2. Persistent storage

9.1.3. Core

[Interface changes. Changes to shared in-core data structures.]

9.2. Installation

[How the component is delivered and installed.]

10. References

[References to all external documents (specifications, architecture and requirements documents, *etc.*) are placed here. The rest of the document cites references from this section. Use Google Docs bookmarks to link to the references from the main text.]