# *<component-name>*

## *Technical Standard Document*

## *version 0.0*

# TABLE OF CONTENTS

# 1 Introduction

*This section should provide an high level overview of the entire Common Component Standard (CCS) document. Include a description of the scope of this standard and its intended usage within the scope of an OERA business application. The component's scope should mention items such as the component's relationships with 3rd party external interfaces, to other CCS components, and provide a brief overview of the visible 'characteristics' of the component such as real-time use, security considerations, concurrency of users etc.*

## 1.1 Purpose

*This section should:*

*a. Describe the purpose of this document;*

*b. Specify the intended readership of this document.*

## 1.2 Scope

*This section should:*

*a. Identify the component being produced;*

*b. Explain what the proposed component will do (and will not do, if necessary);*

*c. Define relevant benefits, objectives and goals as precisely as possible;*

*d. Define any security risks associated with the system;*

*e. Be consistent with similar statements in higher-level specifications, if they exist.*

## 1.3 Definitions, Acronyms and Abbreviations

*This section should define all terms, acronyms and abbreviations used in this document. Particular care should be taken to define terms that are specific to the component, an OERA architecture, application developer, and component developer.*

## 1.4 References to external documents

*If this component standard references information found in other documents this section should list them, identified by title, author and date. Each document should be marked as applicable or reference. If appropriate, report number, journal name and publishing organisation should be included.*

| Num. | Title (Applicability & Reference) | Author | Date | Issue |
|---|---|---|---|---|
| | | | | |
| | | | | |

## 1.5 Contents Overview

*Section 1 is the introduction and includes a description of the project, applicable and reference documents.*

*Section 2 provides the component's overview.*

*Section 3 contains the context in which the component will be applied.*

*Section 4 describes the component's design method, standards and conventions.*

*Section 5 contains the component's component descriptions.*

*Section 6 includes the component's revision history, outstanding issues, and action items*

# 2 Component Overview

*This section should briefly introduce this Common Component Standard's functionality, who the consumers of this component are, and discuss the benefits of why this component exists. This section may also summarise the costs and benefits of the component's selected architecture.*

## 2.1 Component Description

*A description of the component should be given in terms of the architecture that is being implemented and optimally include high level data flows that sets the context of how this component interacts within an OERA architecture as a whole. This section should also set out to 'characterise' the component by describing aspects of its operation that indicate if the system has, inter alia:*

- *to operate in real-time or in bursts, linked to month-end reporting, for example*
- *the nature of the interface to the users of the system*
- *a large number of concurrent users*
- *to be highly resilient or fault tolerant*
- *to provide security features to protect data*
- *to be scaleable and easily maintainable in the future*
- *to have any special back-up facilities to protect important data.*

## 2.2 Component Architecture

*This section should describe the component's architecture, based where feasible on the* OpenEdge Reference Architecture Guidelines*.*

*In this section create a sub-section for:*

- *A description of each concrete class visible to the application developer and provide a high level overview of its role(s), responsibilities, and its relationships with any other classes and/or interfaces*
- *A description of each interface visible to the application developer and provide a high level overview of what type of public functionality and data is exposed to the application developer*
- *A description of the OOABL exception classes used to communicate failure information to the application developer from each and every implementation – the common OOABL methodology should be that every interface method is assumed to complete successfully if an exception is not thrown*
- *A description of the policies and guidance for extending common concrete classes and interfaces that allows added value above what is specified in the standard interfaces*
- *Sub-sections for any other higher level architectural information that will aid an implementer or application developer*

## 2.3 Component Package Definition

In this section provide the fully qualified OO package path the component's classes and interfaces will be a member of

## 2.4 Component Property Data and Organization

*In this section provide a clear and concise description of:*

- *Public OpenEdge datatypes exposed in the class and interfaces*
- *Data naming conventions*
- *External data storage organization requirements ( if any )*
- *ABL Enums*
- *Any data collections used by an interface and how that collection's data values are presented to an application developer*

## 2.5 Component Run-time Characteristics

*In this section provide all information relative to the run-time behaviour as seen by the business application developer. Run-time behaviour topics would include topics such as sequence diagrams, state diagrams, and data flows. The content of this section should provide an unqualified description of the expected run-time behaviour that must be met by an implementer.*

## 2.6 Component Error Handling

*In this section define the strategy and implementation guidance needed to consistently communicate error/exception conditions to the application developer. Include information such as when exceptions are thrown, and how an application developer should interpret and use that information.*

## 2.6 Dependencies and interactions with other OERA common standards

*In this section provide a high level description of all implicit dependencies, explicit dependencies, and interactions with other OERA components.*

# 3 Component Interaction with External Sub-systems

*This section should define the component's use of the external interfaces of other components that are defined in section 2.3. This discussion should be based on a system block diagram or context diagram to illustrate the relationship between this component and other component. The external interfaces should be defined in terms of:*

a. *Types of data being passed*

b. *The form the passed data (i.e. encoding )*

c. *The means of connecting to and using the interface, such as in-process, socket, message system, etc*

d. *Errors raised by the external interface and how this component will handle them*

# 4 Component Interfaces and Classes

*This section should provide a detailed definition of the component's application facing interfaces and concrete classes, including exceptions, used by a business application developer. Each interface and class definition should be contained in its own level-two sub-section, with each sub-section including subsections for:*

a. *Public class [data] properties, per property*
   - *Description of what the data is and will be used for*
   - *Data type*
   - *Get/set*
   - *Any constant values*
   - *Unknown is supported*
   - *Initial default value*

b. *Public class methods, per method*
   - *Description of what action(s) the method performs*
   - *Static classification*
   - *Return data type*
   - *Input, output, input-output parameters, per parameter*
     - *Parameter name*
     - *Input/output/input-output*
     - *Data type*
     - *Valid value range*
     - *Unknown is supported*
     - *Added comments*
   - *Exceptions thrown by the implementation*

# 5 Asynchronous Application Callbacks

*This section should define all of the application facing callbacks that may be registered by the application developer and called by this component. The application callbacks should be defined in terms of:*

a. *When to use the callback*

b. *The types of data being passed between the component and the callback class/procedure*

c. *The form the passed data (i.e. encoding, copy, references, … )*

d. *The direction in which the data is passed*

e. *How the component will handle errors and exceptions raised by the business application code and provide information of how the component's handling of errors and exceptions will/may affect the application*

# 6 Component Design and Implementation

*This and the following section should provide sufficient information for a developer to produce the component. The detailed content will depend upon the approach to the design process that is to be used.*

## 6.1 Design Method and Standards

*The design method used should be named and referenced. A brief description may be added to aid readers not familiar with the method. Any deviations and extensions of the method should be explained and justified. For example if this is a component whose implementation must use OO techniques, this section details which OO techniques may, or may not, be used.*

## 6.1 Naming conventions

*This section should explain all naming conventions used, and draw attention to any points a maintenance programmer would not expect.*

*Conventions for naming files, programs, modules, and possibly other structures such as variables and messages, should all be documented here.*

## 6.3 Programming Standards

*This section should define the project programming standards.*

*Where there are external interfaces, the programming standards for the interfaces required should be referenced.*

*In general, the programming standard should define a consistent and uniform programming style. Specific points to cover are:*

a. Modularity and structuring;

b. Headers and commenting;

c. Indenting and layout;

d. Library routines to be used;

e. Language constructs to use;

f. Language constructs to avoid.

# Document Control

**Title:** OERA <project-name> Standard

**Version:** 1.0

# Document History

| Date | Version | Author | Change Details |
|---|---|---|---|

## Outstanding Issues

*Provide details of any design issues that remain unresolved at the date of issue of this document. Explain options, pros and cons, and give an estimate of which option is most likely. Outline impact of each option on the rest of the design.*