UN/CEFACT
DRAFT
United Nations Centre for Trade Facilitation and Electronic Business

UN/CEFACT Registry
Implementation Requirements Specification

Second Working Draft

The UN/CEFACT Information Content Management Group (ICG) is making this document available for general public review.

ALL COMMENTS TO BE SENT TO: francis.berthomieu@orange-ft.com

EDITORS:

Name	**Affiliation**
Ivan Bedini	France Telecom
Fabrice Bourge	France Telecom

CONTRIBUTORS:

Name	**Affiliation**
Francis Berthomieu	France Telecom
Mary Kay Blantz	UN/CEFACT TMG
Gilles Brandel	SRCI
Edwin Buchinski	Treasury Board of Canada
Colin Clark	UN/CEFACT TBG14
Michael Conroy	Atos Origin (UN/CEFACT ICG)
Eric Delacour	France Telecom
David Dobbing	Exonomy
Thierry Drege	Axway
Fabrice Jeanne	France Telecom
Farrukh Najmi	Sun Microsystems
Duane Nickull	Adobe
Nita Sharma	UN/CEFACT TBG 14
Sébastien Wafflart	France Telecom

Version: 1

Release: 0

Date: June 30, 2006

Foreword

This document contains information as guidance for establishing the UN/CEFACT Registry based on the OASIS ebXML registry specifications.

This document is a Second Working Draft of the UN/CEFACT Registry Implementation Requirements Specification 1.0. This second working draft was created by refining the first working draft according to Step 4 of the UN/CEFACT's Open Development Process (ODP).

Table of Contents

FOREWORD **2**
TABLE OF CONTENTS **3**
ILLUSTRATION INDEX **7**
TABLE INDEX **8**
LISTING INDEX **9**
1 INTRODUCTION **10**
1.1 SUMMARY OF CONTENTS OF DOCUMENT 10
1.2 CHALLENGES 10
1.3 GOALS / REQUIREMENTS / PROBLEM DESCRIPTION 10
1.4 AUDIENCE 11
1.5 SCOPE AND FOCUS 12
1.6 STRUCTURE OF THIS DOCUMENT 12
1.7 DEFINITIONS 13
1.8 TERMINOLOGY 13
1.9 RELATED DOCUMENTS 14
1.10 CAVEATS AND ASSUMPTIONS 14
2 BASIC PRINCIPLES **15**
2.1 OVERVIEW 15
2.1.1 The UN/CEFACT Registry implementation *15*
2.1.2 UN/CEFACT Projects *15*
2.1.3 Operations / Users *16*
2.2 REGISTRY OPERATIONS AND PROCEDURES 16
2.2.1 Registry partitions *16*
2.2.2 Users *17*
2.2.2.1 Guest user 17
2.2.2.2 Registered user 17
2.2.2.3 Administrator 17
2.2.2.4 UN/CEFACT compliant registry 18
2.2.3 Submission procedure *18*
2.2.4 Retrieval procedure *19*
2.2.5 Administration procedure *20*
2.3 LIBRARY PUBLICATION – VERSION/RELEASE MANAGEMENT 21

2.3.1 Core Components versioning ... 21
2.3.1.1 ACC evolutions ... 21
2.3.1.2 ABIE Evolutions ... 23
2.4 SYSTEM DESCRIPTION ... 24
2.4.1 Architecture ... 24
2.4.2 Functional view ... 25
2.4.3 UN/CEFACT Project Artefact ... 27
3 *REGISTRY* STORAGE ... 28
3.1 GLOBAL INFORMATION STORAGE ... 29
3.1.1 Global Context Classification binding to the ebXML RIM ... 32
3.1.1.1 *Registry* Business Context binding to the ebXML RIM ... 38
3.1.2 Registry Registration Information binding to the ebXML RIM ... 40
3.1.2.1 Binding of Registration Information objects to the ebXML RIM ... 40
3.1.2.2 Binding of Registration Information associations ... 43
3.2 CORE COMPONENT STORAGE ... 44
3.2.1 Binding of the CC Information Model to the ebXML RIM ... 46
3.2.1.1 Binding of Core Component objects to the ebXML RIM ... 48
3.2.1.2 Binding of Core Component associations to the ebXML RIM ... 51
3.2.2 Multilingual management ... 52
3.2.3 Version management ... 53
3.2.3.1 Versioning in the [CCTS] ... 54
3.2.3.2 Mapping [CCTS] Version in the *Registry* ... 55
3.3 CODE SET STORAGE ... 57
3.3.1 XML Schemas Directories ... 57
3.3.2 UN/EDIFACT, UN/TDED, UN/LOCODE and related Directories ... 57
3.3.3 Other Directories ... 58
3.4 REFERENCE DOCUMENT STORAGE ... 58
3.4.1 Registry Package ... 58
3.4.1.1 [ebRIM] profile for *Registry* Package ... 59
3.4.2 BRS Artefacts ... 59
3.4.2.1 [ebRIM] profile for BRS ... 59
3.4.3 RSM Artefacts ... 60
3.4.3.1 [eb RIM] profile for RSM ... 61
3.5 BUSINESS PROCESS STORAGE (NOT FOR PROTOTYPE IMPLEMENTATION) ... 62

3.5.1 Business Process Classification ... 62
3.5.1.1 Normative Category ... 62
3.5.1.2 Normative Sub-Category – The ISO Open-edi Phases of a Business Collaboration ... 63
3.5.2 Binding to the ebXML RIM of the BP Classification ... 63
3.5.3 Business Process Information Model ... 64
3.5.4 [ebRIM] profile for CBPC ... 66
4 DATA EXCHANGE FORMAT (INPUT/OUTPUT) ... 67
4.1 REGISTRY EXCHANGE FORMAT ... 68
4.2 BRS AND RSM EXCHANGE FORMAT ... 68
4.3 REGISTRY INPUT FLOW ... 68
4.4 CC EXCHANGE FORMAT ... 69
4.5 CCTS UML MAPPING TO THE EBXML RIM ... 69
4.6 OTHER EXCHANGE FORMATS ... 69
4.6.1 UML/XMI ... 69
4.6.2 Spreadsheet ... 70
4.6.3 NDR-XSD ... 70
5 FEDERATION (NOT FOR PROTOTYPE IMPLEMENTATION) ... 70
5.1 EBXML REGISTRY FEDERATION PRINCIPLE ... 70
5.2 UN/CEFACT REGISTRY FEDERATION ... 71
6 CONFORMITY TESTS (NOT FOR PROTOTYPE IMPLEMENTATION) 73
6.1 CONFORMANCE ... 73
6.2 MANDATORY FEATURES ... 73
6.3 TEST SUITE ... 73
6.4 PROCEDURES FOR TESTING ... 73
6.5 QUALIFIED BODY TO DO TESTING ... 73
7 REGISTRY SERVICES PROFILE ... 73
7.1 CONTENT MANAGEMENT SERVICE PROFILE ... 74
7.1.1 Content Validation Services ... 74
7.1.2 Content Cataloging Services ... 74
7.2 DISCOVERY QUERIES PROFILE ... 75
7.3 EVENT NOTIFICATION PROFILE ... 77
7.3.1 Use Cases for Event Notification ... 77

7.4 SECURITY PROFILE 77
8 APPLICATION INTERFACES 77
8.1 SPECIALIZED INTERFACE 77
8.1.1 Submission Interface 78
8.1.2 Retrieval Interface 79
8.1.2.1 Search functions 79
8.1.2.2 Consultation functions 79
8.1.3 Administration Interface 80
8.2 WEB INTERFACE 81
9 SECURITY 81
9.1 AUTHORIZATION AND ACCESS CONTROL 81
9.2 AUTHENTICATION 82
9.3 INTEGRITY 82
9.3.1 Integrity of Registry Content 82
9.3.2 Integrity of data interchange between the Registry and clients 82
9.4 CONFIDENTIALITY 83
9.5 KEY DISTRIBUTION 83
9.6 SINGLE SIGN ON (SSO) 83
9.7 AUDIT TRAIL 83
10 GLOSSARY 83
ANNEXES 86
A. *REGISTRY* DEPLOYMENT CONSIDERATIONS (INFORMATIVE) 86
B. REPXML OVERVIEW – EXAMPLE (INFORMATIVE) 88
CONTACT INFORMATION 95
COPYRIGHT STATEMENT 96

Illustration Index

Figure 1 - *Registry* conceptual areas 17
Figure 2 - Submission procedure use case diagram 18
Figure 3 - Retrieval procedure use case diagram 19
Figure 4 - Administration procedure use case diagram.....20
Figure 5 - UN/CEFACT simple client/server and interfaces architecture 24
Figure 6 - *Registry* functional and operational view 26
Figure 7 - UN/CEFACT Project Artefact stored in the *Registry* 28
Figure 8 – Global *Registry* storage concept view 30
Figure 9 - Registry view of context classification and registration information 31
Figure 10 - Global Context Classification full definition.....34
Figure 11 - Example of [ebRIM] classification mechanism.....35
Figure 12- *Registry* Registration Information UML information model full definition 40
Figure 13 - Example (not exhaustive) of Financial Account Core Component with Data Types and CC Types 45
Figure 14 - Example (not exhaustive) of Financial Account Business Information Entity for a European Union context 46
Figure 15 - Core Component *Registry* storage UML class diagram full definition.....47
Figure 16 - *Registry* Package content.....58
Figure 17 - Phases of a Business Collaboration.....63
Figure 18 - Abstract Data Structure for the CBPC Information Model 65
Figure 19 - Data exchange principle 68
Figure 20 - Registry federation 70
Figure 21 - UN/CEFACT *Registry* federation architecture (example) 71
Figure 22 - example of a federated query to the *Registry* 72
Figure 23 - ebXML Registry services features.....74
Figure 24 - *Registry* client interface options 78
Figure 25 - RepXML functional architecture 90
Figure 26 - Submit Model use case.....92
Figure 27 - Validate Business Specification use case 93
Figure 28 - Stored Person Business Specification in the RepXML registry 94

Table Index

Table 1 - Definitions....13
Table 2 - Defined [ebRIM] RegistryObject Status Types....29
Table 3 – Abstract *Registry* Class mapping to the ebXML RIM....32
Table 4 - Registry Class relationships binding to the ebXML RIM....32
Table 5 - Classification profile for the *Registry*....36
Table 6 - *Registry* Registration Information profile....41
Table 7 - Registration Information attributes binding to the ebXML RIM....42
Table 8 - *Registry* Registration Information UML associations binding to the ebXML RIM....43
Table 9 - Registration Information UML association classes binding to the ebXML RIM....44
Table 10 - Binding of Core Components to [ebRIM] objects and association types...48
Table 11 - Binding of Core Component classes attributes to the ebXML RIM....51
Table 12 - binding to the ebXML RIM of the Core Components associations....52
Table 13 - Object change version rules....56
Table 14 - [ebRIM] profile for BRS....60
Table 15 - [ebRIM] profile for RSM....61
Table 16 - CBPC information model mapping to the ebXML RIM....67
Table 17 - *Registry* discovery profile for Core Components....76

Listing Index

Listing 1 - Example of *Registry* Product context category expressed in [ebRIM] 38

Listing 2 - Example of [ebRIM] extrinsicObject classification scheme extension for Business Context object types.....39

Listing 3 - Example of [ebRIM] Association class instance for a Replacement Information.....44

Listing 4 - Example of an [ebRIM] Association Class instance.....52

Listing 5 - Mapping to [ebRIM] Classification of the Normative Category (not exaustive) 64

1 Introduction

1.1 Summary of contents of document

This document specifies the criteria for implementing the UN/CEFACT Registry (hereafter referred to as the *Registry*) that will serve as a global registry/repository of reference data for e-business. This document describes at a service level the three main procedures involved:

- The submission procedure – how business data is submitted and validated,
- The retrieval procedure – how business data is accessed and retrieved,
- The administration procedure – how stored business data is managed.

1.2 Challenges

Traditional EDI is largely based on the normalisation of business data exchanges between partners and on the availability of official directories that provide this data. The widespread use of the Internet and related technologies is replacing traditional EDI as the medium of choice for supporting data exchange and for addressing new e-business requirements. In particular ebXML brings high expectations in its ability to go beyond traditional EDI to fulfil new needs, such as the Internet, Web services, etc. However the need to have access to standardised business information remains.

The *Registry* is a global directory where standardised cross-sector business specification data is made available to all. Furthermore, this data provides reusability in the form of Core Components (CC), Business Processes (BP) identified in the Business Requirements Specifications (BRS) and other UN/CEFACT artefacts used in meeting specific business needs.

The *Registry* is the prime mechanism for storing, managing and distributing UN/CEFACT deliverables. However, in the case of CCs and BPs it is assumed and foreseen that there will be eventually many ebXML registries across the world, for example to satisfy the specific needs of a geographical region, a business community, etc. To foster interoperability in e-business exchanges, it is crucial that these ebXML registries be able to federate with the *Registry*.

1.3 Goals / requirements / problem description

The availability of reusable standardised components to enterprises, governmental agencies and/or other organisations is a key element for the development of e-business.

The *Registry* supports the libraries defined by specifications such as the [CCTS] and the [CBPC][1].

The goal of this document is to specify the implementation and operational procedures of the *Registry*. This specification includes:

1 See Section 1.9 (Related documents)

- Definition of the documents that are the objects of the exchange between users of the system,
- Use cases describing the use of the *Registry*,
- Mapping between the data models defined in the [CCTS] and the ebXML registry information model (RIM) defined in the [ebRIM][2],
- Mapping BPs to the RIM defined in the [ebRIM],
- Transformations that are required in order to support automatic submission and retrieval of *Registry* artefacts such as CCs, Business Information Entities (BIEs[3]), BPs, and more generically, Business Requirements Specification (BRS) and Requirement Specification Mapping (RSM) content,
- Use Cases of the workflow application that supports authorised UN/CEFACT working groups to perform various operations on the *Registry* such as submitting, searching, harmonizing, validating and auditing business data according to the procedure approved by UN/CEFACT,
- Description of interfaces: for users, for the client application, and for an external workflow engine.

The *Registry* complies with the ebXML registry specifications (i.e. the [ebRS] and the [ebRIM]).

Also it is important to keep in mind that the *Registry* SHOULD facilitate users, namely the UN/CEFACT Permanent Groups, with their submission procedure (submission and validation of business artefacts) and the business users in their search and retrieval procedure.

Finally, it is essential to be able to retrieve from the *Registry* artefacts that are independent from any syntax, even though the *Registry* may contain business information bound to a specific syntax (e.g. XML, EDIFACT, etc.).

1.4 Audience

The target audience for this document includes all users that will access the *Registry* to search and retrieve one or more BRSs, and any related artefacts such as CCs and BPs underpinning the definition of a business requirement.

The audience includes the various working groups within the UN/CEFACT International Trade and Business Processes Group (TBG), the UN/CEFACT Applied Technology Group (ATG) and the UN/CEFACT Information Content Management Group (ICG) since they will perform several operations, such as submitting, searching, harmonising, validating and auditing business data and processes, using the support services provided by the *Registry*.

The target audience also includes software developers to enable them to put into place a UN/CEFACT compliant registry.

2 See Section 1.9 (Related documents)

3 See Chapter 10 (Glossary)

1.5 Scope and focus

The scope of this document covers the actual *Registry* services and information model as well as the management of roles and responsibilities related to the process of creating *Registry* artefacts in conformance with the Operating Procedures adopted by TBG, ICG and ATG.

This document focuses on registry services that are used to:

- Submit and store UN/CEFACT artefacts (e.g. data and process models, standards, documentation, etc.),
- Discover and retrieve the UN/CEFACT approved artefacts,
- Manage the status of these artefacts, and
- Authorise user access privileges and associated administrative functions.

1.6 Structure of this document

This document gives a detailed view of all aspects of the *Registry* system.

Chapter 1 introduces the objectives, the scope, and the basis on which this document is constructed.

Chapter 2 gives an overview of what the *Registry* service is and how it is used, through several use cases describing the different procedures where the actors submit, retrieve and/or administer UN/CEFACT artefacts. It also provides the basic principles of the release management of the UN/CEFACT deliverables.

Chapter 3 focuses on the storage of the different artefacts within the *Registry*. In particular it defines the ebXML RIM profiles for these artefacts. Section 3.5 - Business process storage (not for prototype implementation) - is not for implementation at this time; its content will be further developed in later versions of this specification.

Chapter 4 defines the input and output data exchange formats.

Chapter 5 provides an insight into what will be the *Registry* federation. This chapter is not for implementation at this time; its content will be further developed in later versions of this specification.

Chapter 6 defines the conformity tests that the *Registry* implementation MUST pass to support this specification, and that all other registry implementations MUST pass to claim compliance to the *Registry* federation. This chapter is not for implementation at this time; its content will be further developed in later versions of this specification.

Chapter 0 defines the services that are provided by the *Registry*, based on the ebXML registry services that are specified in [ebRS].

Chapter 8 defines the methods and applications interfaces that are to be implemented to maintain the *Registry* functionalities.

Chapter 9 covers security aspects of the *Registry* including those that MUST be implemented by UN/CEFACT compliant registry applications.

Chapter 10 provides a glossary of the terms and concepts that are used throughout this document.

Annexes A and B are informative and provide additional information to assist with implementation.

1.7 Definitions

Audit	An official examination and verification of UN/CEFACT official documents by the ICG (in the role of an independent party) to ensure that all procedures and documentation requirements have been respected.
Artefact	Any object seen from the perspective of a *Registry* end user (e.g. BP, CC, code list, etc.).
Object	An Artefact stored in the *Registry*.
Project	A UN/CEFACT Project is a coherent set of artefacts produced by a UN/CEFACT recognized authoritative source that contain business requirements and their generic solutions, based on CCs and common BPs.
Registry	When the word “Registry” is in italics it means the "UN/CEFACT Registry implementation" as specified in this document.
***Registry* Entry**	An instance of an Object.
***Registry* Package**	A *Registry* Package is an object that enables the *Registry* to manage UN/CEFACT Projects. Every instance of a UN/CEFACT Project MUST correspond to a *Registry* Package instance.
Repository	The repository is the central place where UN/CEFACT artefacts are stored and maintained.
Validation	The examination of a given document or object for compliance to defined rules
Verification	The action of establishing or testing the accuracy or correctness of a *Registry* Artefact

Table 1 - Definitions

1.8 Terminology

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in Internet Engineering Task Force (IETF) Request For Comments (RFC) 2119.1

1.9 Related documents

The following documents serve as references:

- [ebRS]: ebXML Registry Services and Protocols, version 3.0,
- [ebRIM]: ebXML Registry Information Model, version 3.0,
- [BRS]: UN/CEFACT Business Requirements Specification, version 1.5,
- [RSM]: UN/CEFACT Requirement Specification Mapping, version 1.0,
- [OP]: UN/CEFACT Operating Procedures between the TBG, ATG & ICG, version 2.1,
- [CCTS]: UN/CEFACT Core Components Technical Specification - Part 8 of the ebXML Framework, version 2.01
- [NDR]: UN/CEFACT XML Naming and Design Rules Technical Specification, version 1.1,
- [BPSS] UN/CEFACT ebXML Business Process Specification Schema v1.01,
- [CBPC]: UN/CEFACT Common Business Process Catalog Technical Specification , version 0.96,
- [UMM] UN/CEFACT UMM Meta Model – Foundation Module. (Candidate for 1.0 - First Working Draft)
- ebXML Technical Architecture Specification, version 1.04,
- ISO 11179: Information technology - Metadata registries,
- RepXML Project (v1.0 - 13 September 2004 - France Telecom).
- [ebRR-UML-TUT] ebXML Registry Tutorial: UML to ebRIM mapping
- [ebRR-DPT] Deployment Profile Template For ebXML Registry 3.0 OASIS Specifications
- [WSS-SMS]: Web Service Security: SOAP Message Security 1.0.

1.10 Caveats and assumptions

To thoroughly understand the content of this document, it is recommended that the reader have a minimum knowledge of the [UMM] as well as the [CCTS].

This document provides a specification on how to implement the *Registry*. It also provides the basis for a future standard document specifying the implementation of any CC and BP registry that may be federated to the *Registry*.

2 Basic principles

2.1 Overview

2.1.1 The UN/CEFACT *Registry* implementation

The *Registry* provides a persistent store where reference information is registered and published in order to facilitate e-business entity and process design. It is based on the [ebRS] and [ebRIM] ebXML registry standards that specify the constructs of an information system that securely manages any content type, the standardized metadata that describes it, and the set of services that enable the registry information to be shared among the registry clients.

Artefacts are stored as Objects in the *Registry* and managed by the *Registry* as defined in the [ebRS].

The [ebRIM] provides information on the type of metadata that is stored in an ebXML registry as well as the relationships amongst metadata classes. The [ebRIM] defines in a generic manner what types of objects are stored and defines how these objects are organized. Examples and tutorial information may be found at the following web site: http://www.oasis-open.org.

2.1.2 UN/CEFACT Projects

An ebXML registry can store any type of content, e.g. CCs, classification code lists, BPs, collaboration profiles and agreements, XML schema and documents, UML models, general documents, etc.

The *Registry* MAY eventually reference and store many content types. Recognising that a primary concern of UN/CEFACT is the publication of business requirements and their generic solutions, based on CCs and common BPs, this specification focuses on the registration and publication of any type of Artefact related to these requirements. Typically an Artefact corresponds to a UN/CEFACT Project which consists of one or more of the following deliverables:

- BRS,
- BP,
- RSM,
- CC and BIE (in the sense of what is defined in the [CCTS]),
- Technical solution.

A UN/CEFACT Project is subject to the operating procedures adopted by the ATG, ICG and TBG as defined in the [OP].

On the other hand, an Artefact MAY also correspond to a single piece of information from a UN/CEFACT recognized authoritative source. For example, an ISO code list may be submitted by an authorised user for direct registration and publication on the *Registry*.

2.1.3 Operations / Users

All operations that can be performed on the *Registry* fall into four categories:

- Information submission,
- Information validation,
- Information access and retrieval,
- *Registry* administration.

Information MAY only be submitted to the *Registry* by authorised submitting organisations. These submitting organisations are principally the UN/CEFACT Permanent Groups, namely the TBG, the ICG and the ATG. Other submitting organisations MAY include any external organisations duly authorised by UN/CEFACT.

Information validation MAY result in a change of status of *Registry Entries*. This is performed by authorised users designated by the TBG, ATG and/or ICG.

Information access and retrieval covers information search (at a basic level), consultation and retrieval. User access will depend on the status and the ownership of the objects stored in the *Registry*.

2.2 Registry operations and procedures

The *Registry* is a system that references and stores artefacts for e-business. Depending on requirements, various operations MAY be performed on the *Registry* by defined types of users. For example, the operating procedures adopted by the ATG, ICG and TBG in developing the various artefacts will involve workflow through direct interaction with the *Registry*. Up until the artefacts have attained a status of "published", the maintenance of the artefacts will be the responsibility of the UN/CEFACT Permanent Groups and as such will be subject to access control. Once published the artefacts are openly available for access by all users.

2.2.1 *Registry* partitions

The *Registry* provides a means for maintaining the status of an Artefact. The status lifecycle can be represented by four conceptual areas as shown in Figure 1:

- The Working area where users with specific access rights (registered users) can use the *Registry* to store and share content having a "draft" status,
- The Validation area where draft artefacts have been submitted for approval, designated with a "submitted" status,[2]
- The Pre-publication area where artefacts have received an "approved" status and are ready to be audited before publication,
- The Publication area where artefacts have received a "published" status and are then accessible by all users.

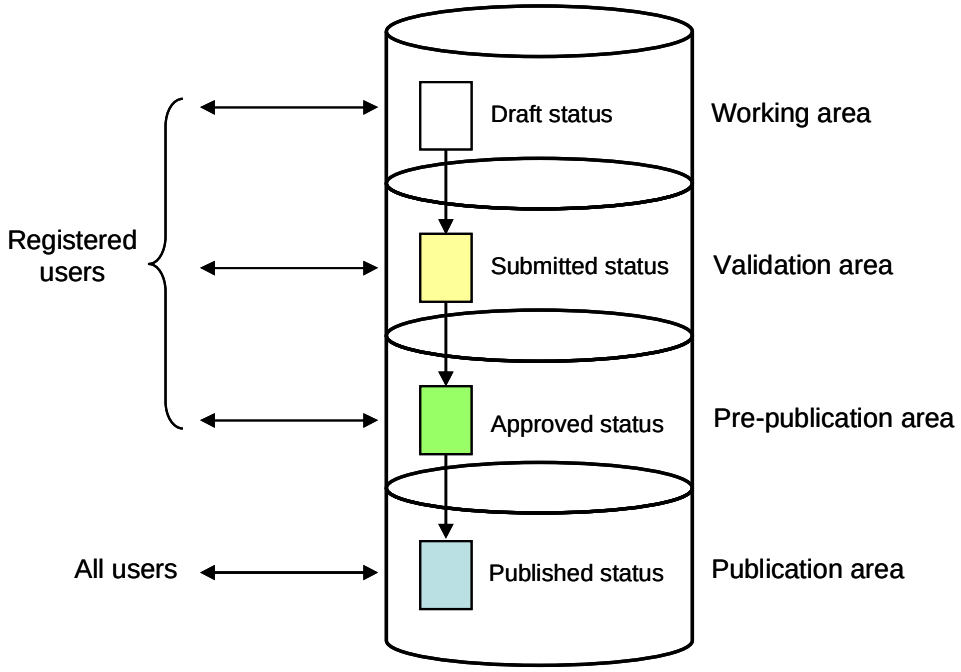

Figure 1 - *Registry* conceptual areas

2.2.2 Users

Actors interacting with the *Registry* correspond to the roles defined in the [ebRS]; they can be individual users or software programs that interact with the *Registry* using ebXML registry protocols.

2.2.2.1 Guest user

A guest user is a user that can use the *Registry* without being registered. Consequently the identity of a guest user is not known to the *Registry*. Therefore a guest user has limited privileges within the *Registry*. For example, a guest user can only access artefacts that are publicly available.

2.2.2.2 Registered user

A registered user is a user that is registered to the *Registry*. Consequently the identity of a registered user is known to the *Registry*, allowing him/her to have privileged access to the *Registry*. For example, a registered user can approve artefacts that are not yet publicly available.

2.2.2.3 Administrator

An administrator is a registered user that has privileged access rights that allow him/her to perform the administrative tasks necessary for the ongoing operation of the *Registry*. Such operations include, for example, modifying the access rights of a registered user.

2.2.2.4 UN/CEFACT compliant registry

A UN/CEFACT compliant registry is one that complies with the specific set of features described in Chapter 6 of this specification. Furthermore, a certificate of compliance MUST be obtained from UN/CEFACT to claim such compliance.

2.2.3 Submission procedure

The submission procedure comprises three use cases which are depicted in Figure 2.

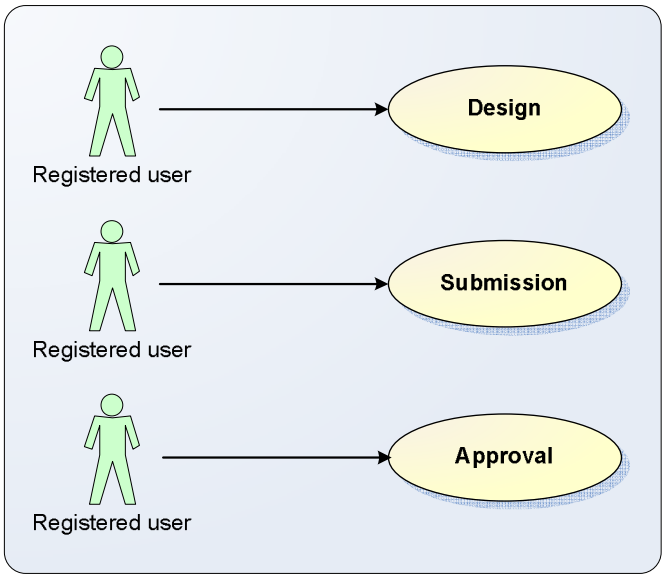

Figure 2 - Submission procedure use case diagram

The *design* use case describes the situation where a registered user wishes to use the *Registry* during the design phase of a new *Project*. In this case draft Artefacts MAY be stored within the *Registry* before being submitted for approval; however they do not have to follow all the *Registry* content validation constraints. The *Registry* Entry created by the *Registry* to catalogue the draft Artefact takes the status "draft" (see Table 2 for possible status values).

The *submission* use case describes the situation where a registered user submits an Artefact to the *Registry* for approval. This may happen, for example, upon completion of the design phase depicted above. The submitted Artefact can be a Project or any other Artefact. Only registered users that have appropriate access rights can perform this submission operation. The *Registry* Entry corresponding to the submitted Artefact takes the status "submitted".

The *approval* use case describes the situation where a registered user approves an Artefact that has previously been submitted to the *Registry*. Only registered users that have appropriate access rights can perform this approval operation. Approval can only apply to an Artefact for which the *Registry* Entry has a "submitted" status; if the approval operation succeeds, then the corresponding *Registry* Entry takes the status "approved". In the case where the newly approved (version of the) Artefact supersedes a previous version of the same Artefact for which the *Registry* Entry status

was "approved", then the status of the latter MUST be changed to "deprecated" (see Table 2).

For example, after being harmonized with existing repository items by TBG 17, the BRS must be validated by the TBG as a whole prior to the ODP public review stage. Upon approval of the draft BRS, the status of the corresponding *Registry* Entry changes from "submitted" to "approved".

Submission and/or approval of artefacts can be done individually or in a batch mode.

2.2.4 Retrieval procedure

The retrieval procedure comprises two use cases which are depicted in Figure 3.

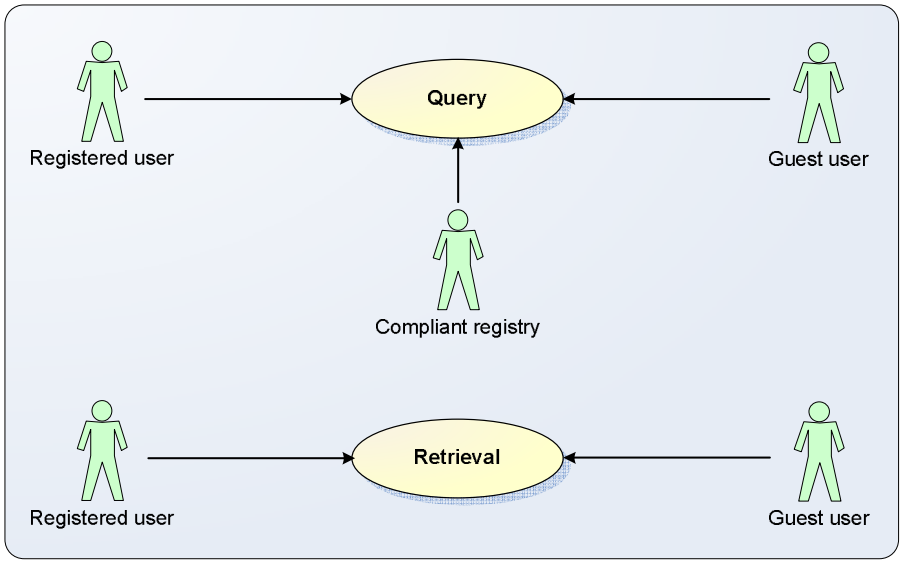

Figure 3 - Retrieval procedure use case diagram

The *Query* use case describes the situation where a registered user, a guest user or a UN/CEFACT compliant registry submits a query to the *Registry* to search and retrieve one or several artefacts. Guest users and compliant registries can only query Artefacts that are publicly available, i.e. Artefacts for which the corresponding *RegistryObjects* have the "approved" and/or the "deprecated" statuses. Registered users can query, in addition to all public artefacts, all non public artefacts for which they have the ownership, and possibly some non public artefacts for which they do not have ownership but for which they have access rights (at least in read-only mode). An Artefact is not publicly available if its corresponding *RegistryObject* has either the "submitted" status or the "withdrawn" status. However, a registered user can query a non public Artefact only if its *RegistryObject* has the "submitted" status.

For instance, when harmonizing a BRS and/or RSM the project team members of the relevant TBG group would examine existing *Registry* entries to determine the relevance and/or applicability of currently registered BIE and CC instances. They may also query the *Registry* to get all pending BRSs and RSMs that have not yet been approved, in order to harmonize two or more BRSs and/or RSMs that are being

processed concurrently. It is assumed that all members of any TBG group or project team in charge of a BRS/RSM will have the rights to view (at a minimum) all other pending BRSs and RSMs.

The *retrieval* use case describes the situation where a guest user or a registered user simply retrieves an Artefact or a set of artefacts from the *Registry*. This occurs, for example, when the user browses through Web pages and then wishes to download an identified Artefact or a set of artefacts.

2.2.5 Administration procedure

The administration procedure comprises three use cases which are depicted in Figure 4.

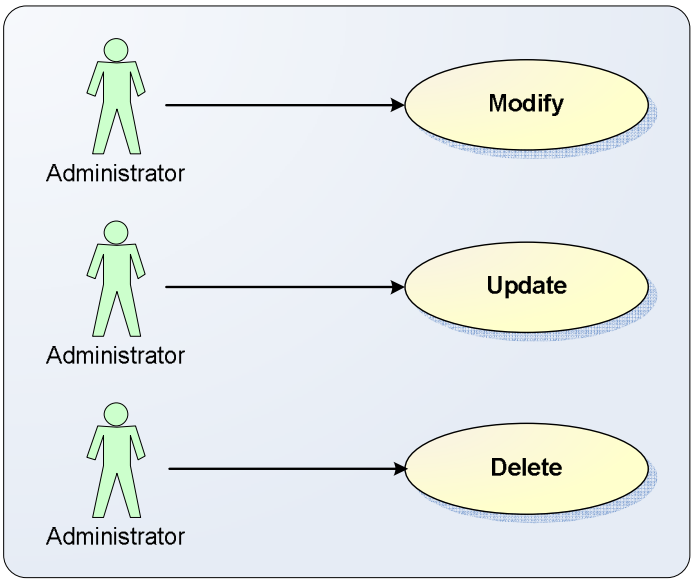

Figure 4 - Administration procedure use case diagram

The *modify* use case describes the situation where the administrator modifies an Artefact. For example, the administrator wants to change the owner of an Artefact or a hierarchical set of artefacts.

The *update* use case describes the situation where a minor revision is made to an Artefact that has reached an "approved" status. The "administrator" role for certain kinds of updates MAY be confined to specific authorities (e.g. TBG 17 which performs a data harmonization function within the UN/CEFACT).

The *delete* use case describes the situation where the administrator deletes an Artefact from the *Registry*. On the one hand if the *Registry* Artefact is unattached (i.e. when it is not used anywhere else), then it is actually deleted. On the other hand when it has previously been approved and deployed in e-Business, its status MUST be set to "deprecated" and the Artefact MUST remain in the *Registry* until it is not linked to any Artefact used in the *Registry*.

2.3 Library publication – Version/release management

The UN/CEFACT FMG has adopted the basic principle of releasing all the UN/CEFACT deliverables that are under FMG responsibility on a twice yearly basis as is currently carried out with the UN/EDIFACT library production. This will ensure that for a given release all the information within it is in complete coherence.

Each release will be published on the UN/CEFACT *Registry* and will be known as a UN/CEFACT Publication release and will be assigned a Publication release number which will be in the form R.yyA and R.yyB where the letter "R" signifies "release" and the letters "yy" are replaced by the last two digits of the year of the publication release. The last letters "A" and "B" indicate the order of the release in the year, the letter "A" release being normally released in the June timeframe and the letter "B" being released in the December timeframe. The contents of each publication release have to be agreed at the preceding UN/CEFACT forums (March and September respectively).

Nothing will prevent at a later date increasing the number of releases through the sequential augmentation of the release letters (A, B, C, D, E, etc.).

The UN/CEFACT *Registry* will also provide the capability of keeping a version number for each Object maintained in the *Registry*. This version number will only increase when the Object in question has evolved between publication releases. The version number SHALL only increment by one for each publication release evolution.

Only the latest version of each Object SHALL be included in a publication release. The UN/CEFACT *Registry* SHALL manage a historical log of all previous Object versions. However within the UN/CEFACT working groups the UN/CEFACT *Registry* SHALL only permit the use of the current publication release Object version.

2.3.1 Core Components versioning

In order for the versioning of the CC Objects to ensure overall consistency, some specific rules SHALL be put into place. These rules are outlined in the following paragraphs.

2.3.1.1 ACC evolutions

An Aggregate Core Component (ACC) represents the controlled vocabulary of a given object class. It will be possible to carry out the following basic functions on the object class itself and its associated Basic Core Components (BCCs):

1. Add a BCC to the list of controlled phrases (ACC attributes. This implies an additional BCC; however all additional BCCs MUST be OPTIONAL.);
2. Change the multiplicity of a BCC from 0..1 to 0..* or from 1 to 1..* or from 1 to 0..* (in this case the objects already based on the existing ACC are remain coherent since it is merely increasing the constraint during use);
3. Change the description of an ACC or a BCC;
4. Add a new ASCC that has a conditional relationship with other ACC object classes;
5. Add a mandatory BCC (uniquely if all object classes that are based on the ACC are simultaneously changed to add the BCC in question);
6. Add a mandatory ASCC (uniquely if all object classes that are based on the ACC are simultaneously changed to add the ASCC in question).

It is not allowed to:

1. Change the name of the ACC (this implies the creation of a new ACC);
2. Change the name of a BCC (this implies adding a BCC to the existing ACC);
3. Delete a BCC (this implies modifying the multiplicity from 1 or 1..* to 0..1 or 0..*);
4. Add a mandatory BCC without ensuring that all associated object classes that are based on the ACC will be changed simultaneously to add the equivalent BCC in question (This implies modifying all associated objects before such a change can take place);
5. Delete an ASCC (this implies modifying the multiplicity from 1 or 1..* to 0..1 or 0..*);
6. Add a mandatory ASCC without ensuring that all associated object classes that are based on the ACC will be changed simultaneously to add the equivalent ASCC in question (this implies modifying all associated objects before such a change can take place).

A change of the ACC version number SHALL occur whenever there is a modification of the type mentioned in points 1, 2, 4, 5 and 6 of the permitted operations mentioned above.

A change of the ACC revision number SHALL occur whenever there is a modification of the type mentioned in point 3 above since this does not imply any structural change to the ACC in question.

All version values SHALL be numeric values that SHALL begin with 1 and SHALL increment by 1 with each publication release where one or several changes to the structure of the object in question as outlined above is published.

All release values SHALL be numeric values that SHALL begin with 0 for each new version number and SHALL increment by 1 with each publication release where there is only one or several changes to the description of the object in question or its components as outlined above is published.

An ACC SHALL be available in the working part of the UN/CEFACT Registry prior to publication to facilitate development.

All object classes that are based on an ACC SHALL not be impacted by its evolution. In cases where such an impact can occur then all the relative objects involved in the change MUST be evolved in the same publication release. This covers such objects as ABIEs, Electronic documents, RSMs, etc.

The RSM document, which is generally associated with a BRS, is the means where a new ACC or an evolution to an ACC is submitted. This document is the transmission vehicle between the TBG groups and the harmonisation process and the UN/CEFACT Permanent Groups that are concerned by ACC use or publication. This document is a UN/CEFACT internal working document that SHALL be used for the submission of object classes and for the specification of the mapping between the Conceptual business model and the UN/CEFACT Canonical model.

2.3.1.2 ABIE Evolutions

An Aggregate Business Information Entity (ABIE) MUST always be based on an ACC. This implies that in the RSM document that is used for the submission of the ABIE one of the following conditions is met:

1. The ABIE is based on a new ACC in which case the ACC is completely specified and the ABIE contextual and constraining information are in the same document;
2. The ABIE is based on an existing ACC that requires no change, in which case the ACC is referenced and the ABIE contextual and constraining information is provided;
3. The ABIE is based on an existing ACC that requires some changes, in which case the ACC is referenced, the changes to the ACC are provided and the ABIE contextual and constraining information is provided;
4. The ABIE makes use of an existing ABIE in which case the ABIE in question is merely referenced;
5. The ABIE is based on an existing ACC that requires some changes that impact other ABIES, in which case the ACC is referenced, the changes to the ACC are provided, the ABIE contextual and constraining information is provided and the complete list of impacted ABIEs is also provided (note: this will require the signed off agreement of all the concerned groups agreeing to the change both to the ACC and to all related documentation.

It is not possible for an ABIE to be based on an existing ABIE where there are modifications to the ABIE in question. This is the same as the creation of a new ABIE.

The following evolutions are permitted for an existing ABIE:

1. Add a new BBIE (provided it already exists in the ACC and providing that any other uses of the ABIE in question are modified at the same time);
2. Change the ABIE or a BBIE description;
3. Add new context categories;

4. Delete context categories;
5. Replace context categories;
6. Change the BBIE multiplicity from 0..1 to 1 (provided that any other uses of the ABIE in question are modified at the same time);
7. Change the BBIE multiplicity from 1 to 0..1.

The following evolutions are not permitted:

1. add new qualifiers to the ABIE name (this implies adding a new ABIE);
2. Add new qualifiers to a BBIE name (this implies adding a new BBIE).

2.4 System Description

2.4.1 Architecture

This section describes the Architecture of the *Registry* implementation.

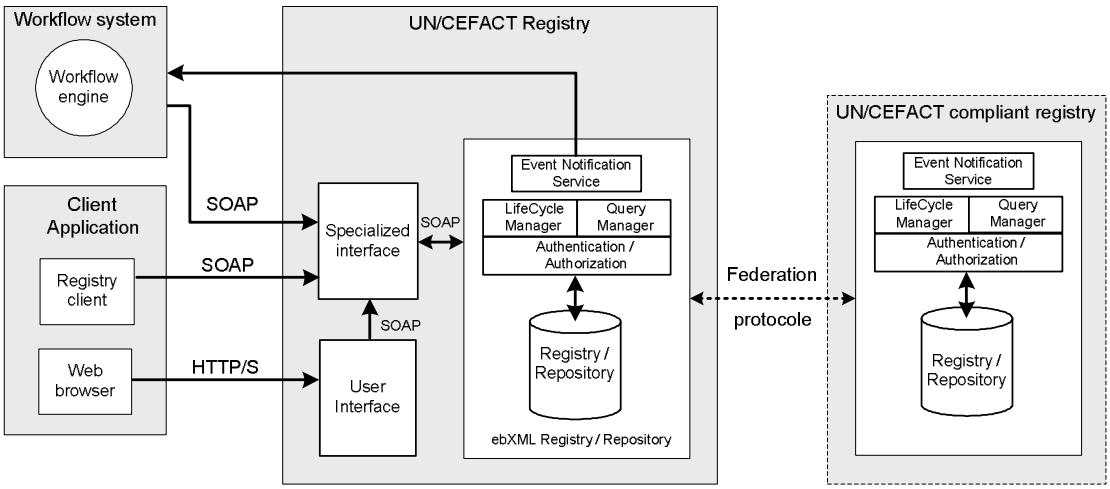

Figure 5 - UN/CEFACT simple client/server and interfaces architecture

UN/CEFACT *Registry*:

The *Registry* is an application that allows and simplifies the standardization process and the publication of business specifications.

Workflow system:

This holds the external workflow engine that supports *Registry* interface procedures. This engine SHALL provide an API capable of interfacing with the *Registry* in order to receive notifications from the event notification service. Such a system MAY also make requests to the *Registry* in place of the client application.

Web interface:

A Web browser provides Internet based access to the *Registry* content.

Specialized interface:

The specialized interface supports user and application interaction with UN/CEFACT Artefacts, e.g. through modelling tools, and provides methods to control the structure, semantic and syntax rules for submitted Artefacts. The Specialized interface interacts with the ebXML registry/repository through the JAXR (Java API for XML Registries) API.

Service interface protocol binding (HTTP / SOAP):

The *Registry* allows access and management of UN/CEFACT Artefacts by web interfaces and SOAP with attachment methods.

ebXML registry/repository:

This is an ebXML registry/repository application that is integrated with the *Registry*.

Client application:

The client application allows users to manage *Registry* Artefacts through the following types of clients:

> ***Registry* client:** A *Registry* client could be a "fat client" that implements a secure *Registry* interface to submit, validate and retrieve *Registry* Artefacts.
>
> The *Registry* client could be implemented as a dedicated application or as a part of a more generic tool (for example a UML modelling tool could generate specific XML formats that could be submitted to the *Registry* by using a web service supported by a specialized *Registry* interface).
>
> **Web browser:** The Web browser can be used as an alternative to the *Registry* Client for the administration and management of *Registry* Artefacts and is the preferred utility to allow users to discover and retrieve *Registry* Artefacts.

UN/CEFACT compliant registry:

A UN/CEFACT compliant registry is an external registry that MUST support the *Registry* information model as well as the *Registry* federation protocol. All UN/CEFACT Compliant Registries MUST have passed the relevant *Registry* conformity tests (refer to Chapter 6).

2.4.2 Functional view

Figure 6 provides an overview of the *Registry* functions that facilitate the harmonization activities of the UN/CEFACT permanent groups.

The harmonization activities performed by the UN/CEFACT Permanent Groups are as follows:

- TBG assures the submission in a correct format and manages the request submission of authorized external organizations. TBG harmonizes and validates UN/CEFACT Artefacts before their publication.
- ICG manages the release of and access to Artefacts developed and/or certified by UN/CEFACT Working Groups.
- ATG provides technical solutions to the submitted requests and verifies the compliance of auto-generated technical solutions where necessary.

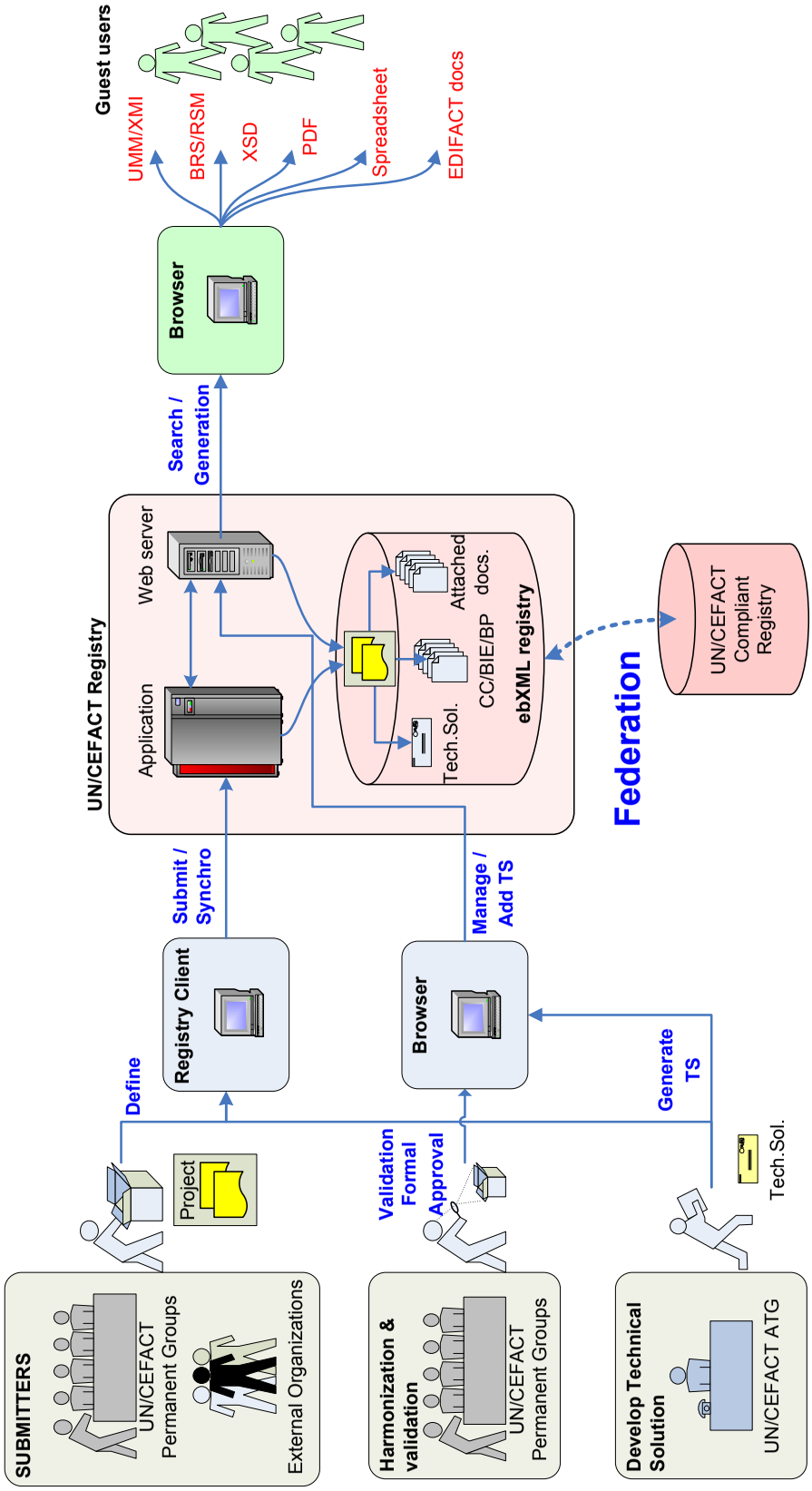

Figure 6 - *Registry* functional and operational view

The harmonization activities performed by the UN/CEFACT Permanent Groups are as follows:

- TBG assures the submission in a correct format and manages the request submission of authorized external organizations. TBG harmonizes and validates UN/CEFACT Artefacts before their publication.
- ICG manages the release of and access to Artefacts developed and/or certified by UN/CEFACT Working Groups.
- ATG provides technical solutions to the submitted requests and verifies the compliance of auto-generated technical solutions where necessary.

The *Registry* provides interfaces for submission and can accept several input formats. It provides a multi-criteria search function for registry/repository content. It provides some auto generated solutions for known output formats (e.g. XMI, XSD for BIEs…).

Guest users can discover public *Registry* content by using search tools that support the *Registry* specialized interface. Depending on the situation the content MAY be retrieved as stored in the *Registry* or it MAY be dynamically generated into a defined specific *Registry* format. For example, the UMM class diagram of an ABIE may be dynamically generated into an XMI format.

2.4.3 UN/CEFACT Project Artefact

Fundamentally a UN/CEFACT Project consists of multiple Artefacts which will be developed, assembled and submitted in accordance with *Registry* prescribed formats. For this reason every Artefact MUST be transformed into a defined XML format and MUST be mapped to the target storage format, the RIM, by the *Registry* application (see Figure 7).

The following is a non exhaustive list of the Artefacts that can comprise a UN/CEFACT Project and how they should be mapped to the RIM:

- the BRS SHOULD be transformed in an XML file corresponding to a DTD with one or more attached UMM schemas in XMI format, or to another recognized format, that correspond to the related BPs. This complete set is known as the BRS Artefact,
- the RSM, as for the BRS, SHOULD be submitted in an XML file format, or to another recognized format, corresponding to a predefined DTD with one or more UMM class diagrams of the related BIEs. This complete set is known as the RSM Artefact,
- Electronic documents MAY be any form and type of document considered relevant for the completeness of the business specification. These documents normally will not be considered for research criteria,
- Technical solutions developed by ATG SHOULD be added to the Project as attached descriptive electronic documents.

Note that the exact definition of XML, DTD and XSD constructs is outside the scope of this specification.

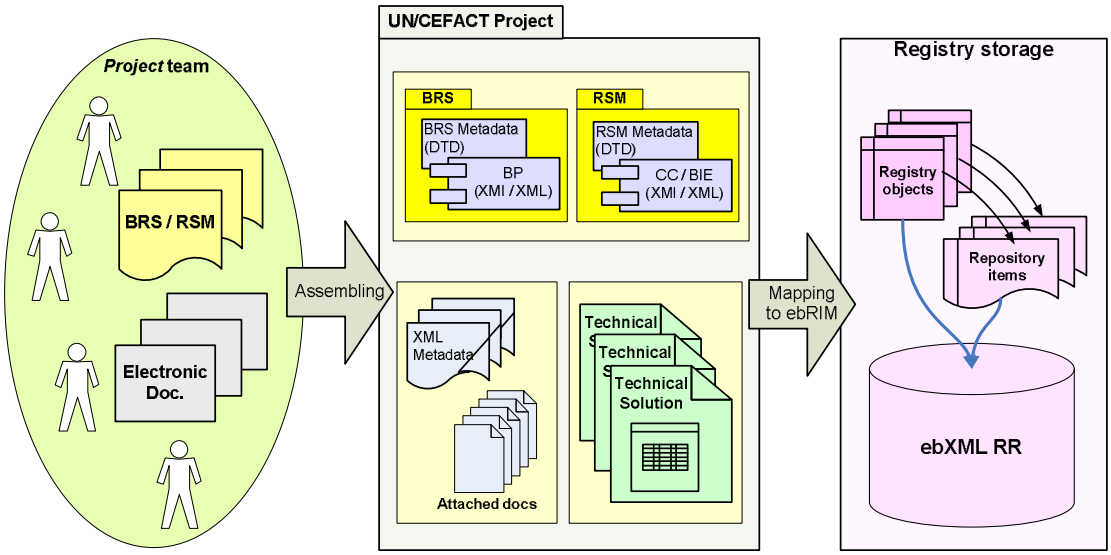

Figure 7 - UN/CEFACT Project Artefact stored in the *Registry*

3 *Registry* storage

This chapter defines:

- all UN/CEFACT Artefacts that can be stored in the *Registry*,
- the structure of the UN/CEFACT Artefacts,
- the mapping of UN/CEFACT Artefacts to the RIM,
- general rules for cataloguing the UN/CEFACT Artefacts to support subsequent identification and discovery,
- guidelines for supporting validation of UN/CEFACT Artefacts.

All identified UN/CEFACT Artefacts SHALL be mapped into the RIM, the reference model for all objects that are stored and managed by the *Registry*[4].

Accurate mapping to the RIM is considered fundamental for *Registry* compliant implementations. Identical mapping across distributed registry implementations is REQUIRED to support registry federation and to share *Registry* Entries.

The *Registry* will contain all UN/CEFACT approved and submitted business specifications, related technology solutions and associated data or files.

The associated data can be XML schemas, EDIFACT data, XForms, ASN1 data and any other data.

The principle is that all approved Artefacts contained in the *Registry* MAY be retrieved and re-used by any business group.

4 Throughout this section the UN/CEFACT Artefacts are considered as the source concept of the mapping and the RIM as the target model.

Every Artefact is stored within the *Registry* as a *RegistryObject* which can take one of the five status attribute values as defined in Table 2. The capability of creating additional status types MUST be catered for.

Name	Description
Draft	Stored RegistryObject that has a status of "draft", for example for working draft documents
Submitted	Stored RegistryObject that has a status of "submitted", for example for validation/approval
Approved	Stored RegistryObject that has a status of "approved"; it is has passed the UN/CEFACT validation and approval process
Published	Stored RegistryObject that has a status of "published"; it is has been successfully audited and can be viewed by all guest users
Deprecated	Stored RegistryObject that has a status of "deprecated"; it will be replaced or removed at a later date
Withdrawn	Stored RegistryObject that has a status of "withdrawn". A *Registry* item has been removed but its ExtrinsicObject still exists

Table 2 - Defined [ebRIM] RegistryObject Status Types

Approved UN/CEFACT Artefacts that are stored in the *Registry* MAY be retrieved and re-used by anyone without any access restrictions. Submitted Artefacts MAY be retrieved by the submitter group, the validation group or any other authorized person.

3.1 Global information storage

A global registry is a global container for several different concepts, such as BPs, messages, business information, CPP/CPA templates, etc. As a global registry the UN/CEFACT *Registry* MUST be able to store every type of e-business content and also, where possible, it SHOULD provide a standard common set of *Registry* Entries for accommodating all administrative information and classifications that facilitate transversal content discovery and storage.

This section illustrates the concept of a common *Registry* storage mechanism. Figure 8 illustrates the conceptual view of UN/CEFACT Artefact storage. This storage mechanism MAY be viewed as specifying the artefact stored in the *Registry*, its registration information and its context classification.

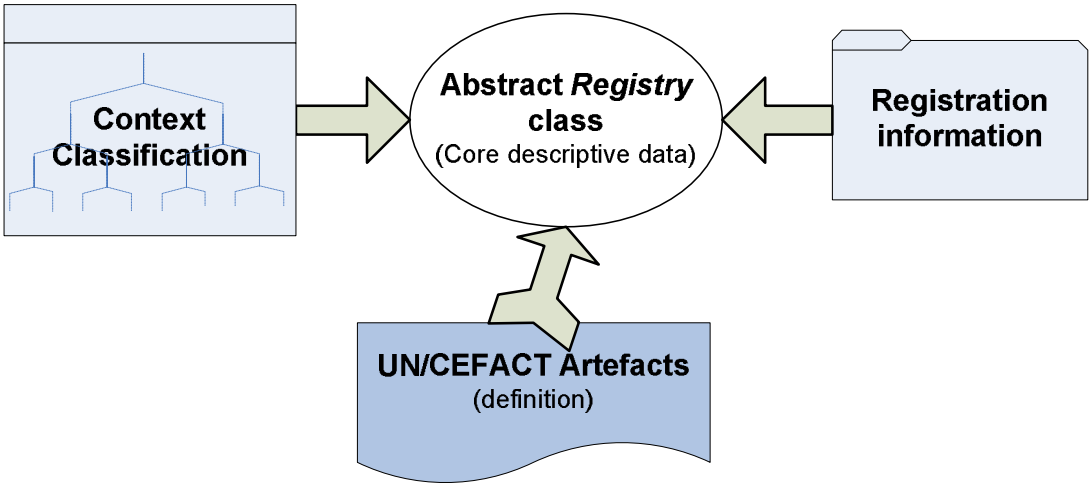

Figure 8 – Global *Registry* storage concept view

The main concepts of the global *Registry* view are as follows:

- The Abstract *Registry* class is the root of the different concepts of this view. It collects the core descriptive generic data required by all stored *Registry* Entries (e.g., Name, Description, Version, etc.). Every *Registry* Entry MUST have all mandatory data defined in this class with related semantics.
- The Context Classification concept represents the common semantics used for identifying the terms qualifying the business usage of Artefacts. Adapting this common view to all Artefacts contained within the *Registry* enables a starting point for discovery. From the end users perspective it is advantageous to discover specific content through its business usage and find all pertinent Artefacts for a specific context.
- The Registration Information concept includes all the administrative information, such as the owner, the additional documentation, the history, etc. of a *Registry* Entry. Every UN/CEFACT Artefact MUST have this information registered and accessible to users.
- The UN/CEFACT Artefact concept corresponds to the source Artefact information model to be stored. For example the CC and BIE information models are specific Artefacts stored within the *Registry*. All UN/CEFACT Artefacts inherit from the Abstract *Registry* class.

Figure 9 illustrates the integrated model for context classification and registration information that is more fully described in the subsections that follow. Every stored UN/CEFACT Artefact SHOULD be classified and registered, and SHOULD inherit attribute metadata from the abstract *Registry* Class as indicated in this model.

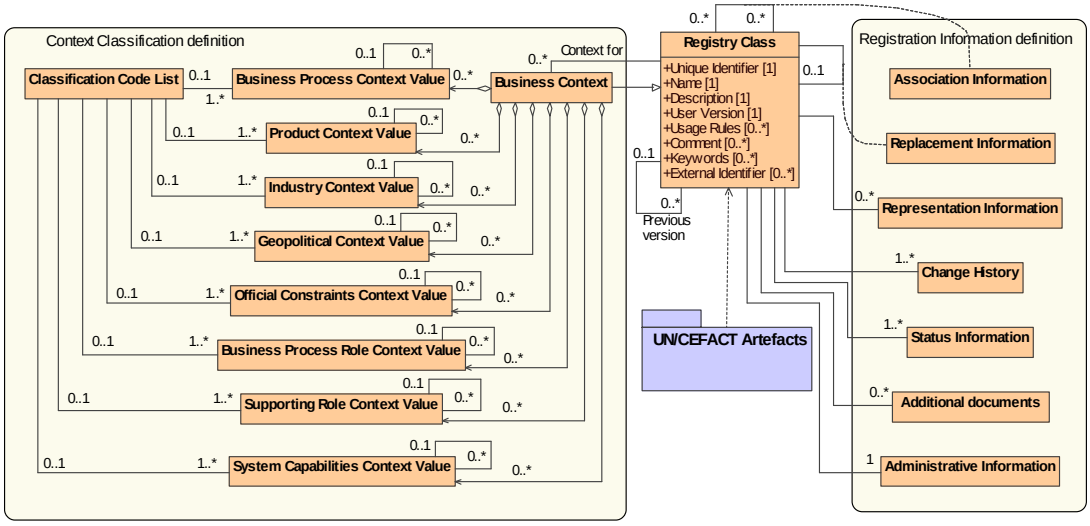

Figure 9 - Registry view of context classification and registration information

Table 3 shows the whole mapping of the Registry Class to the RIM and has the following information:

- *Source Attribute*. The source attribute item represents the attribute of the source concept that MUST be mapped to an already existing RIM Registry Entry attribute or to a new slot instance defined here;
- *[ebRIM] Attribute*. The definition of the mapped source attribute. Slots attributes are defined as follows: Slot(*name, type, 'value'*);
    - *Name* is the corresponding name for the slot (for example: *usageRule);*
    - *Type* is one of the permitted RIM types for a *slotType* as defined in [ebRIM];
    - *Value* can be a list of permitted values for this attribute that SHOULD be verified by the *Registry* content validation service at the submission or 'any value' if no constraints exists for the attribute. (for example an attribute can admit only 'Yes' or 'No' values);
- *Cardinality*. It represents the number of distinct values that this attribute can contain. A particular exception is for attributes with cardinality equal to 1 as max value of instances that are mapped to Registry Entry attributes that provide a multilingual value, such as name and description for example, only one value for each language MUST be accepted;
- *Comment*. Optionally, any other additional comment.

The Registry Class attributes defined SHOULD be inherited by all UN/CEFACT Artefacts instances.

Source Attribute	[ebRIM] Attribute	Card.	Comment
Source concept: **Registry Class**			
Unique Identifier	Registry Entry Unique Identifier	1..1	If no ID is provided by the submitter, the default system UUID registry identifier is used. [ebRIM] logical id attribute MAY have the same value.
External Identifier	ExternalIdentifier	0..*	
Name	Name	1..1	Canonical *Registry* Entry name attribute. The Name could be used as a part of the Logical ID registry attribute.
Description	description	1..1	Canonical *Registry* Entry description attribute.
User Version	VersionComment	0..1	The *Registry* Entry attribute versionInfo is the official version for the *Registry* Entry. Any user version value used for other purposes can be added here. Same thing for repository item, if any. User version value can be mapped to ContentVersionComment attribute.
Usage Rule	Slot (usageRule, String, any string)	0..*	
Keywords	Slot (keywords, String, any string)	0..*	
Comment	Slot (comment, String, any string)	0..*	

Table 3 – Abstract *Registry* Class mapping to the ebXML RIM

Table 4 shows the Registry Class relationships binding to the RIM.

Association Source Object	Association Target Object	[ebRIM] Association Type	[ebRIM] Association Name
Registry Class	Business Context	ContextFor	-

Table 4 - Registry Class relationships binding to the ebXML RIM

3.1.1 Global Context Classification binding to the ebXML RIM

Figure 10 identifies the global context classification method that has been designed to provide the semantic qualification of UN/CEFACT Artefacts. This method allows users to uniquely identify and distinguish all contained business concepts within different Business Contexts.

Every Artefact can be classified by adding one or more associations with one or more business contexts. A *Registry* Business Context instance is then defined as a unique and meaningful combination of context values. UN/CEFACT currently has eight context categories that can be used to classify and discover all stored *Registry* Artefacts. These are[5]:

5 From [CCTS] §6.2.2 – Approved Context Categories

- *Business Process Context* – This is the classification of the BP, business collaboration or business transaction as described in the [CBPC]. It is the primary context category, and provides many useful distinctions in the analysis of CCs.
- *Product Classification Context* – There are many types of information that are specific to products or services being traded or referred to in a BP.
- *Industry Classification Context* – Traditionally, business vocabularies are divided into industry verticals. This Context Category specifies a particular industry vertical.
- *Geopolitical Context* – Specifies the semantic and structural variation. This is often the result of regional or cultural factors.
- *Official Constraints Context* – Specifies the legal or contractual influences upon Business Semantics.
- *Business Process Role Context* – Every partner in a BP data exchange has a particular role – buyer, seller, etc. These roles are described in the [CBPC] and in other Business Libraries (libraries of BP models). Depending on the BP, the nature of these roles may require that certain semantics and data be employed in the messages exchanged. In any Business Process Role Context, one must either be a sender or receiver of data in that particular exchange – otherwise, role is described by the Supporting Role Context.
- *Supporting Role Context* – Parties in a BP who are neither senders nor receivers of data in a particular exchange, may place requirements on the data exchanged by partners who are sending or receiving of data in that exchange. These non-sending, non-receiving parties in this exchange play a supporting role, and are described by the Supporting Role Context.
- *System Capabilities Context* – When a particular semantic or structure is primarily the result of system constraints or compliance with a standard, it is attributable to the System Capabilities Context.

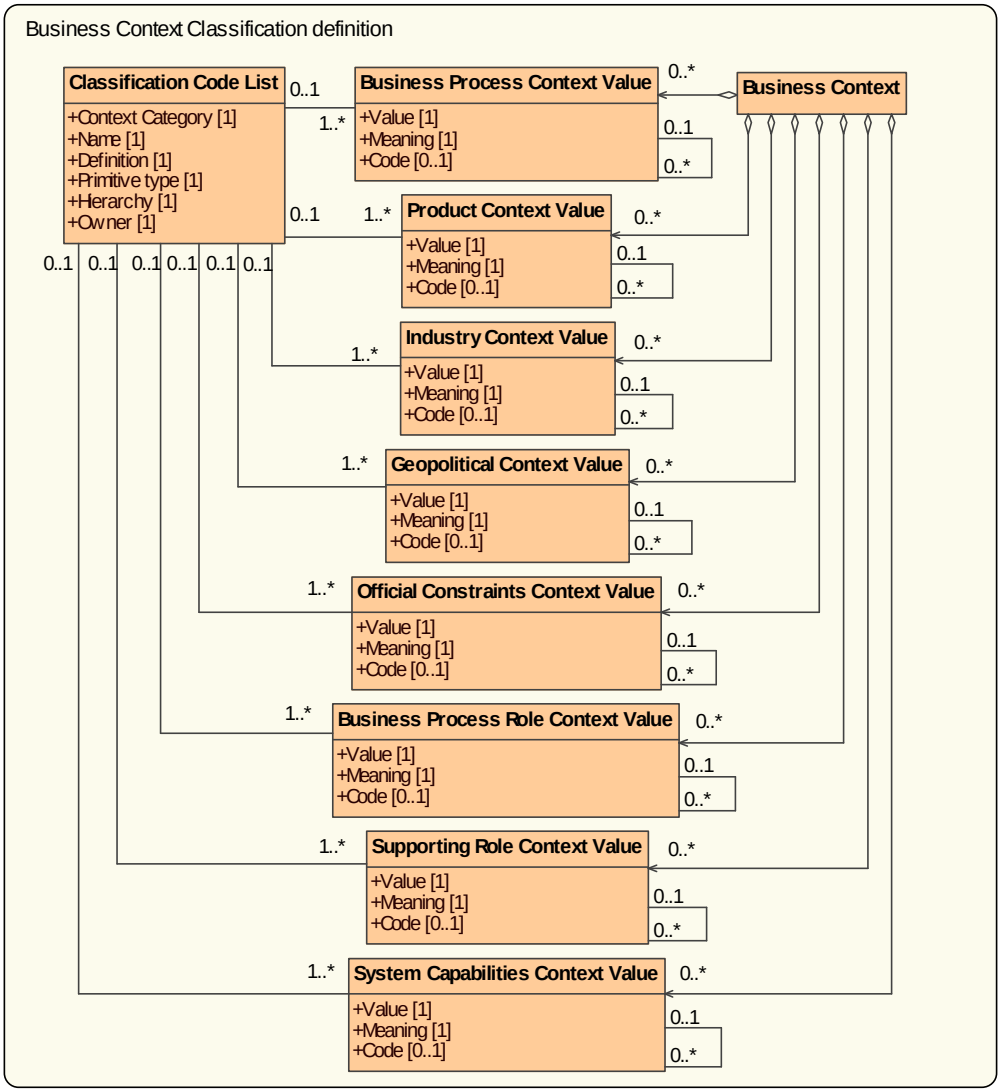

Figure 10 - Global Context Classification full definition

[ebRIM] provides an extensible means of classifying stored object instances within an ebXML registry. Each classification category is added to the RIM as a new [ebRIM] *ClassificationScheme* and the associated classification code list is added as one or more new [ebRIM] *ClassificationNodes*.

The [ebRIM] classification mechanism MAY be used for grouping subsets of context values within one or more Business Context instances. It adopts the tree structure[6] graph where each node has zero or more child nodes. Consequently every code list used for classifying a business context MUST be added within the *Registry* as a *ClassificationNode* under the corresponding *ClassificationScheme* root node containing the corresponding code list values. If several code lists are identified for a *Registry* context category, they SHOULD be added to the tree as sibling *ClassificationNodes* (see Figure 11).

6 See: http://en.wikipedia.org/wiki/Tree_structure for more details

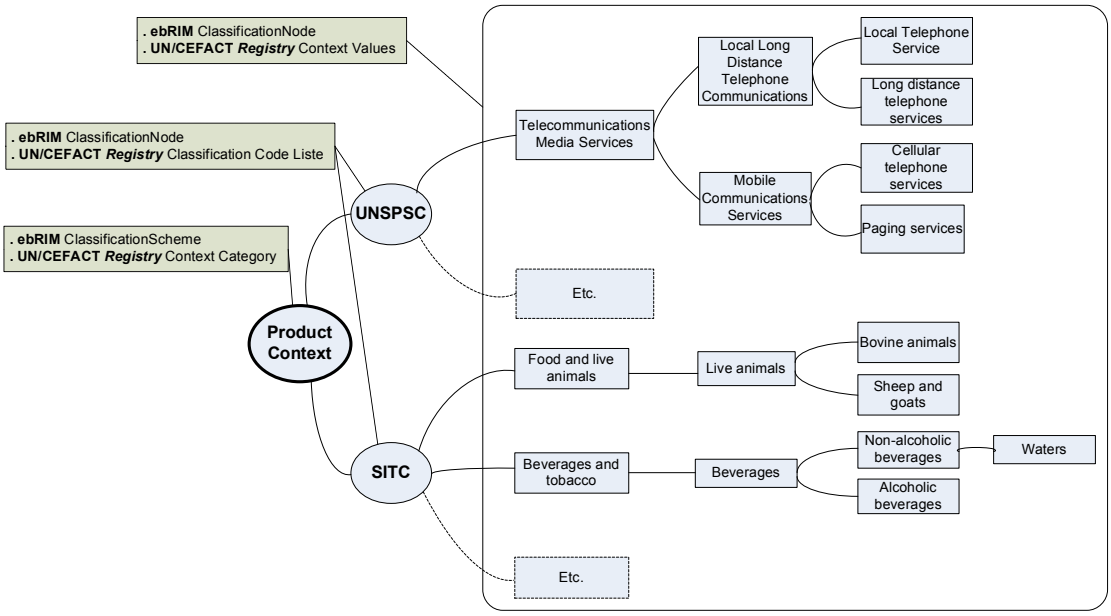

Figure 11 - Example of [ebRIM] classification mechanism

Table 5 defines the [ebRIM] *ClassificationSchemes* that MUST be created to support the eight UN/CEFACT Context Categories. The *Registry* will enforce the roles and responsibilities defined by the maintenance agencies for the list of reference and context values

Source Context Classification Category	[ebRIM] Classification Scheme	Classification Code List	Description
Business Process Context Value	BusinessProcessContext	[CBPC], UMM Business Process Normative Category	The Business Process name(s) as described using the UN/CEFACT Catalogue of Common Business Processes as extended by the user.
Product Context Value	ProductContext	Universal Standard Product and Service Specification (UNSPSC); Standard International Trade Classification (SITC Rev .3); Harmonized Commodity Description and Coding System (HS); Classification Of the purposes of non Profit Institutions serving households (COPI).	Factors influencing semantics that are the result of the goods or services being exchanged, handled, or paid for, etc. (e.g. the buying of consulting services as opposed to materials)
Industry Context Value	IndustryContext	International Standard Industrial Classification (ISIC); Universal Standard Product and Service Specification (UNSPSC) Top-level Segment [digits 1 and 2] used to define industry; NAICS – North American Industry Classification Scheme.	Semantic influences related to the industry or industries of the trading partners (e.g., product identification schemes used in different industries).
Geopolitical Context	GeopoliticalContext	Geopolitical Context values	Geographical factors that

Source Context Classification Category	[ebRIM] Classification Scheme	Classification Code List	Description
Value		shall be taken from ISO 3166.1 and 3166.2	influence Business Semantics (e.g., the structure of an address).
Official Constraints Context Value	OfficialConstraintsContext	No known global classification of all Official Constraints Contexts	Legal and governmental influences on semantics (e.g. hazardous materials information required by law when shipping goods).
Business Process Role Context Value	BusinessProcessRoleContext	UN/EDIFACT Code List for DE 3035 Party Roles	The actors conducting a particular Business Process, as identified in the UN/CEFACT Catalogue of Common Business Processes.
Supporting Role Context Value	SupportingRoleContext	UN/CEFACT Common Business Process Catalogue	Semantic influences related to non-partner roles (e.g., data required by a third-party shipper in an order response going from seller to buyer)
System Capabilities Context Value	SystemCapabilitiesContext	No known classification of all types of information systems and standards.	This Context Category exists to capture the limitations of systems (e.g. an existing back office can only support an address in a certain form).

Table 5 - Classification profile for the *Registry*

It must be noted that the UML class Classification Code List, contained within the Global Context Classification class diagram (Figure 10), is mapped to the RIM as the sub-node of the corresponding Context Category and its sub-node are the context values representing the classification list.

Listing 1 shows the UNSPC standard list of reference, taxonomy, as a classification scheme for the *Registry* Product Context category expressed in the [ebRIM] (the list is not exhaustive):

```
<!-- Defining Product Context - Classification Scheme - with a sub-set of
UNSPSC V8.0401 list -->
<ClassificationScheme lid="Classification:ProductContext"
id="urn:un:unece:uncefact:icg:registry:ContextClassification:ProductContext"
isInternal="true" nodeType="urn:oasis:names:tc:ebxml-
regrep:NodeType:UniqueCode">
<Name><LocalizedString charset="UTF-8" value="Product Context" xml:lang="en-
US"/></Name>
<Description>
<LocalizedString charset="UTF-8" value="Factors influencing semantics that
are the result of the goods or services being exchanged, handled, or paid
for, etc. (e.g. the buying of consulting services as opposed to materials)"
xml:lang="en-US"/>
</Description>
<!-- The ClassificationNode below (UNSPSC80401) is the binding of the
Classification Scheme UML class illustrated in Figure 9 -->
<ClassificationNode code="UNSPSC80401"
parent="urn:un:unece:uncefact:icg:registry:ContextClassification:ProductCont
ext" lid="ContextClassification:ProductContext:UNSPSC-8_0401"
id="${UNSPC_classificationNode_UUID}"
path="/urn:un:unece:uncefact:icg:registry:ContextClassification:Product/UNSP
SC">
<Slot name="owner" slotType="urn:oasis:names:ebXML-regrep:DataType:String">
<!-- either it can be an objectRef to another Registry Entry representing
the owner such as an organization or a person -->
<ValueList><Value>UNSPSC</Value></ValueList>
</Slot>
<Slot name="contextCategory" slotType="urn:oasis:names:ebXML-
regrep:DataType:String">
<ValueList><Value>ProductContext</Value></ValueList>
</Slot>
<Slot name="hierarchy" slotType="urn:oasis:names:ebXML-
regrep:DataType:String">
<ValueList><Value>yes</Value></ValueList>
</Slot>
<Slot name="primitiveType" slotType="urn:oasis:names:ebXML-
regrep:DataType:String">
<ValueList><Value>String</Value></ValueList>
</Slot>
<Name><LocalizedString charset="UTF-8" value="UNSPSC" xml:lang="en-
US"/></Name>
<Description>
<LocalizedString charset="UTF-8" value="Universal Standard Product and
Service Specification" xml:lang="en-US"/>
</Description>
<ClassificationNode code="83110000" lid="TelecommunicationsMediaServices"
id="${TelecommunicationsMediaServices_classificationNode_UUID}"
path="/ContextClassification:Product/UNSPSC/TelecommunicationsMediaServices"
>
<Name><LocalizedString value="Telecommunications Media Services"
xml:lang="en-US"/></Name>
<ClassificationNode code="83111500"
lid="LocalLongDistanceTelephoneCommunications"
id="${LocalLongDistanceTelephoneCommunications_classificationNode_UUID}"
path="/ContextClassification:Product/UNSPSC/TelecommunicationsMediaServices/
LocalLongDistanceTelephoneCommunications">
<Name><LocalizedString value="Local Long Distance Telephone Communications"
xml:lang="en-US"/></Name>
<ClassificationNode code="83111501" lid="LocalTelephoneService"
id="${LocalTelephoneService_classificationNode_UUID}"
path="/ContextClassification:Product/UNSPSC/TelecommunicationsMediaServices/
LocalTelephoneService">
```

```
<Name><LocalizedString value="Local telephone service" xml:lang="en-
US"/></Name>
</ClassificationNode>
<ClassificationNode code="83111502" lid="LongDistanceTelephoneServices"
id="${LongDistanceTelephoneServices_classificationNode_UUID}"
path="/ContextClassification:Product/UNSPSC/TelecommunicationsMediaServices/
LongDistanceTelephoneServices">
<Name><LocalizedString value="Long distance telephone services"
xml:lang="en-US"/></Name>
</ClassificationNode>
</ClassificationNode>
<ClassificationNode code="83111600" lid="MobileCommunicationsServices"
id="${MobileCommunicationsServices_classificationNode_UUID}"
path="/ContextClassification:Product/UNSPSC/TelecommunicationsMediaServices/
MobileCommunicationsServices">
<Name><LocalizedString value="Mobile communications services" xml:lang="en-
US"/></Name>
<ClassificationNode code="83111603" lid="CellularTelephoneServices"
id="${CellularTelephoneServices_classificationNode_UUID}"
path="/ContextClassification:Product/UNSPSC/TelecommunicationsMediaServices/
CellularTelephoneServices">
<Name><LocalizedString value="Cellular telephone services" xml:lang="en-
US"/></Name>
</ClassificationNode>
<ClassificationNode code="83111500" lid="PagingServices"
id="${PagingServices_classificationNode_UUID}"
path="/ContextClassification:Product/UNSPSC/TelecommunicationsMediaServices/
PagingServices">
<Name><LocalizedString value="Paging services" xml:lang="en-US"/></Name>
</ClassificationNode>
</ClassificationNode>
</ClassificationNode>
</ClassificationNode>
</ClassificationScheme>
```

Listing 1 - Example of *Registry* Product context category expressed in [ebRIM]

3.1.1.1 *Registry* Business Context binding to the ebXML RIM

The Business Context concept plays a central role in the discovery process of *Registry* content. It can be the starting point of a user search and the way to grouping *Registry* Entry instances having common aspects on semantic and domain. A drill-down system for the discovery of *Registry* instances based on the content classification SHOULD show only existing business contexts for each context category classifying at least one business context and not the whole list of context values, which in several cases amount to a huge list. Every business context can appear as an element for the category list where it is classified.

An ebXML registry permits the extension of its classification mechanism in order to provide specific content if the defined [ebRIM] canonical Registry Entries do not closely match the source object concept or class. The binding of the Business Context class, as for all other Artefacts, to the RIM is carried out by defining a new sub-node to the canonical [ebRIM] *ExtrinsicObject* object type. Attributes are mapped to the existing canonical *Registry* Entry attributes or, where it does not exist, to an adequate attribute within the *Registry* class, an [ebRIM] Slot object can be added to the

Registry Entry as defined in [ebRIM]7. The Business Context class also inherits the abstract *Registry* Class attributes as illustrated in Figure 9.

The relationship between a Business Context instance and the corresponding set of values of the 8 context categories (defined in Section 0) to classify that instance MUST be realised using the [ebRIM] classification system. Within an ebXML registry classification and taxonomy are called *ClassificationScheme*, the permitted values are represented by the sub-node called *ClassificationNode* and it can be used to classify *RegistryObjects* to facilitate discovery based upon that classification.

Listing 2 shows an example of a [ebRIM] *ExtrinsicObject* for a *Registry* Business Context instance with a factious geographical classification.

```
<!-- Defining Business Context Object Type-->
<ClassificationNode parent="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject"
lid="urn:un:unece:uncefact:icg:registry:ObjectType:RegistryObject:ExtrinsicO
bject:BC" code="BusinessContext" id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:BC"/>
<!—- Example of a Business Context instance using the registry extensibility
feature -->
<ExtrinsicObject id="${BC_french_langage_regions-countries_ID}"
objectType="urn:un:unece:uncefact:icg:registry:ObjectType:RegistryObject:Ext
rinsicObject:BC">
<Name>
<LocalizedString value="BC_country-areas_French_language" xml:lang="en-US"/>
<LocalizedString value="BC_pays-regions_langue_francais" xml:lang="fr-FR"/>
</Name>
<!—Use of the registry classification system to classify the instance -->
<Classification
classificationNode="urn:un:unece:uncefact:icg:registry:ContextClassification
:Geopolitical:iso3166:Europe:FR"
classifiedObject="${BC_french_langage_regions-countries_ID}"
id="${Geopolitical_classification_ID1}"/>
<Classification
classificationNode="urn:un:unece:uncefact:icg:registry:ContextClassification
:Geopolitical:iso3166:NorthAmerica:CA:QC"
classifiedObject="${BC_french_langage_regions-countries_ID}"
id="${Geopolitical_classification_ID2}"/>
</ExtrinsicObject>
```

Listing 2 - Example of [ebRIM] extrinsicObject classification scheme extension for Business Context object types

The specifically defined classification nodes represent the [ebRIM] Registry Entry *objectType* attribute of the *ExtrinsicObject* object that MUST be specified in every submitted instance. This feature provides a good way to personalize the *Registry* implementation. It supports the creation of predefined search and discovery queries (for example: get all *ExtrinsicObject* where objectType is "*urn:un:unece:uncefact:icg:registry:ObjectType:RegistryObject:ExtrinsicObject:BC*") as well as *Registry* content management.

Every [ebRIM] *ExtrinsicObject* instance can have one or more references to a classification node defined within a particular *ClassificationScheme* using directly the [ebRIM] classification element attribute.

7 For more details about the mapping to [ebRIM] method and rules from a source information model the readers can see [ebRR-DPT] and [ebRR-UML-TUT] documents.

The Business Context class is mapped to the RIM as sub-node of [ebRIM] *ExtrinsicObject* classification node and no additional attributes of the Registry Class, as defined in Table 3, are required.

3.1.2 *Registry* Registration Information binding to the ebXML RIM

A wide variety of administrative information concerning the status, provenance and change history of reusable data and business process design Artefacts MAY exist within the *Registry*. This administrative information MUST be searchable and retrievable in order to simplify its discovery and use in business document and process management and revision. For this reason every *Registry* Entry SHOULD have a reference to the registration information.

This section defines the complete binding to the RIM of the Registry Registration Information UML information model illustrated in Figure 12.

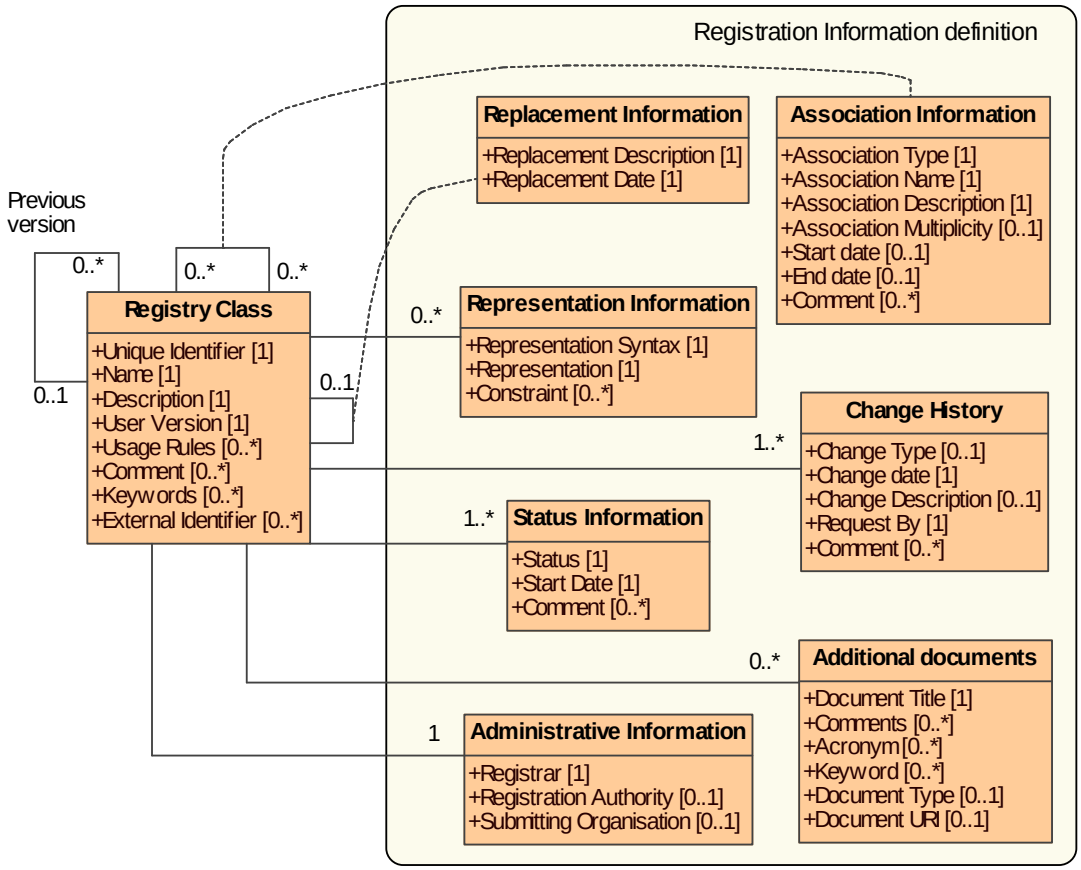

Figure 12- *Registry* Registration Information UML information model full definition

3.1.2.1 Binding of Registration Information objects to the ebXML RIM

This section defines the complete binding of the global Registry Registration information to the RIM. Table 6 lists the mapped UML classes, represented by the Source Concept column, and the corresponding [ebRIM] new classification node.

Source Concept	[ebRIM] Object Type Name	[ebRIM] Parent Type Name	Comment
Change History	ChangeHistory	extrinsicObject	Information about changes that are made to a *Registry* Entry. Some of these changes information are already provided by the registry within the A*uditableEvent.* Implementers of this profile SHOULD use this registry feature rather than create new objects of this type.
Status Information	StatusInformation	extrinsicObject	Information about the live status of [CCTS] artefacts could be mapped directly to the registry status attribute for registry instances. If a new status value is needed by a specific implementation, it MAY added extending the canonical registry status list.
Administrative Information	AdministrativeInformation	extrinsicObject	Information about the registration of [CCTS] artefacts SHOULD be mapped to the Registry Entry canonical attributes, such as owner for the source model registrar attribute, rather than creating a new Registry Entry of this type.
Representation Information	RepresentationInformation	extrinsicObject	Registry Entry instances of this type represent information about the physical representation of a [CCTS] artefact in a particular syntax and they MUST have a repository item corresponding to the description or definition of the particular syntax (eg. the XML schema) or a link to an external accessible resource.
Descriptive Information	DescriptiveInformation	extrinsicObject	Additional descriptive information or document can be associated to the [CCTS] artefacts using this object type.

Table 6 - *Registry* Registration Information profile

Table 7 lists the mapping for attributes of the Registration Information UML class diagram.

Source Attribute	[ebRIM] Attribute	Card.	Comment
Source concept: **Representation Information** (Sub-Node ExtrinsicObject/RepresentationInformation)			
Representation Syntax	Name	1..1	
Representation	description	1..1	
Constraint	Slot (constraint, String, any string)	0..*	
Source concept: **Descriptive Information** (Sub-Node ExtrinsicObject/DescriptiveInformation)			
Reference Document	Name	1..1	
Comment	Slot (comment, String, any string)	0..*	Even if a description is not provided within the information source model. Implementers SHOULD consider to add a description to instances rather than generic comments.
Document Type	Slot (documentType, String, any string)	0..1	
Document URI	ExternalLink	0..1	The document SHOULD be added as the repository item to the Registry Entry instance or as an external link.

Source Attribute	[ebRIM] Attribute	Card.	Comment
Acronym	Slot (acronym, String, any string)	0..*	
Keyword	Slot (keyword, String, any string)	0..*	
Source concept: **Change History** (Sub-Node ExtrinsicObject/ChangeHistory)			
Change Type	Name	0..1	
Change Date	Slot (changeDate, Date, any date)	1..1	
Change Description	description	0..1	
Request By	Slot (requestBy, String, any string)	1..1	It MAY be a reference ([ebRIM] objectRef) to a registered user
Comment	Slot (acronym, String, any string)	0..*	
Source concept: **Status Information** (Sub-Node ExtrinsicObject/StatusInformation)			
Status	name	1..1	The entire Status information class SHOULD be mapped directly to the [ebRIM] auditable and the status could be mapped to the status registry attribute object. If it doesn't match the implementation requirement, the StatusInformation [ebRIM] extrinsic object MUST be used.
Start Date	Slot (changeDate, Date, any date)	1..1	If status informations are mapped to the autitable events the start date is assumed as the date where the Registry Entry has updated its status.
Comment	description	0..*	
Source concept: **Administrative Information** (Sub-Node ExtrinsicObject/AdministrativeInformation)			
Registrar	Slot (registrar, String, any string)	1..1	Implementers SHOULD consider that each *Registry* Entry has an owner, which is a registry registered user. The owner is often the registrar. In this case this slot can be omitted. If the submission has been done using the registry cataloguing service the *Registry* Entry owner (from the registry auditable event) is a reference to the service itself. In this case this slot MAY contain a reference ([ebRIM] objectRef) to the registered registry user that has submitted the object.
Registration Authority	Slot (registrationAuthority, String, any string)	0..1	Depending to implementation issues this slot MAY refer to a registry registered organization that for the *Registry* is the UN/CEFACT.
Submitting Organisation	Slot (submittingOrganization, String, any string)	0..1	Depending to implementation issues this slot MAY be implicitly the belongings organization of the submitter or MAY refer to a registry registered organization, if any. For the *Registry* it is often one of the UN/CEFACT permanents groups.

Table 7 - Registration Information attributes binding to the ebXML RIM

3.1.2.2 Binding of Registration Information associations

The [ebRIM] considers associations as Registry Entries and this feature permits the mapping of the UML association class directly to [ebRIM] associations. Replacement Information and Association Information association classes can be mapped directly to the registry association object and class attributes mapped to the canonical registry association object attributes or added as slots. Table 8 defines the binding to the RIM of the UML associations illustrated in Figure 12.

Association Source Object	Association Target Object	[ebRIM] Association Type	[ebRIM] Association Name
Registry Class	Replacement Information	ReplacedBy	
Registry Class	Association Information	AssociatedTo	*${Association_Information_Name}*
Registry Class	Change History	RelatedTo (canonical [ebRIM] ass. type)	*${linkedRegistryClassName_ChangeType}*
Registry Class	Representation Information	RelatedTo (canonical [ebRIM] ass. type)	
Registry Class	Status Information	RelatedTo (canonical [ebRIM] ass. type)	
Registry Class	Administrative Information	RelatedTo (canonical [ebRIM] ass. type)	
Registry Class	Descriptive Information	RelatedTo (canonical [ebRIM] ass. type)	

Table 8 - *Registry* Registration Information UML associations binding to the ebXML RIM

Table 9 lists the association classes and the corresponding attributes binding to the RIM.

Source Attribute	[ebRIM] Attribute	Card.	Comment
Source concept: **Replacement Information** (Sub-Node AssociationType/ReplacedBy)			
Replacement Description	Description	1..1	
Replacement Date	Slot (replacementDate, Date, *any date*)	1..1	
Source concept: **Association Information** (Sub-Node AssociationType/AssociatedTo)			
Association Name	Name	1..1	
Association Type	Slot (type, String, *any string)*	1..1	
Association Description	description	1..1	
Association Multiplicity	Slot (multiplicity, String, *any string)*	0..1	
Start Date	Slot (startDate, Date, *any date)*	0..1	
End Date	Slot (endDate, Date, *any date)*	0..1	
Comment	Slot (comment, String, *any string)*	0..*	

Table 9 - Registration Information UML association classes binding to the ebXML RIM

Listing 3 shows an example of a Replacement Information association class instance for the ACC Account that has been replaced by the Financial Account ACC.

```
<rim:Association id="${ASSOCIATION_IDENTIFIER_ID}"
associationType="urn:oasis:names:tc:ebxml-regrep:AssociationType:ReplacedBy"
sourceObject="${SOURCE_Account_ACC_IDENTIFIER_ID}"
targetObject="${TARGET_Financial_Account_ACC_IDENTIFIER_ID}">
<rim:Description>
<rim:LocalizedString value="Discovery of a duplication object concept"/>
</rim:Description>
<rim:Slot name="replacementDate" slotType="urn:oasis:names:ebXML-
regrep:DataType:Date">
<ValueList><Value>herethedate</Value></ValueList>
</rim:Slot>
</rim:Association>
```

Listing 3 - Example of [ebRIM] Association class instance for a Replacement Information.

3.2 Core Component storage

The Core Component is the central concept; it is a building block for the creation of a semantically correct and meaningful information exchange package. It contains only the information pieces necessary to describe a specific concept. The example

illustrated in Figure 13 shows the four different categories of CCs, these are: Basic CC, Association CC, CC Type and Aggregate CC.

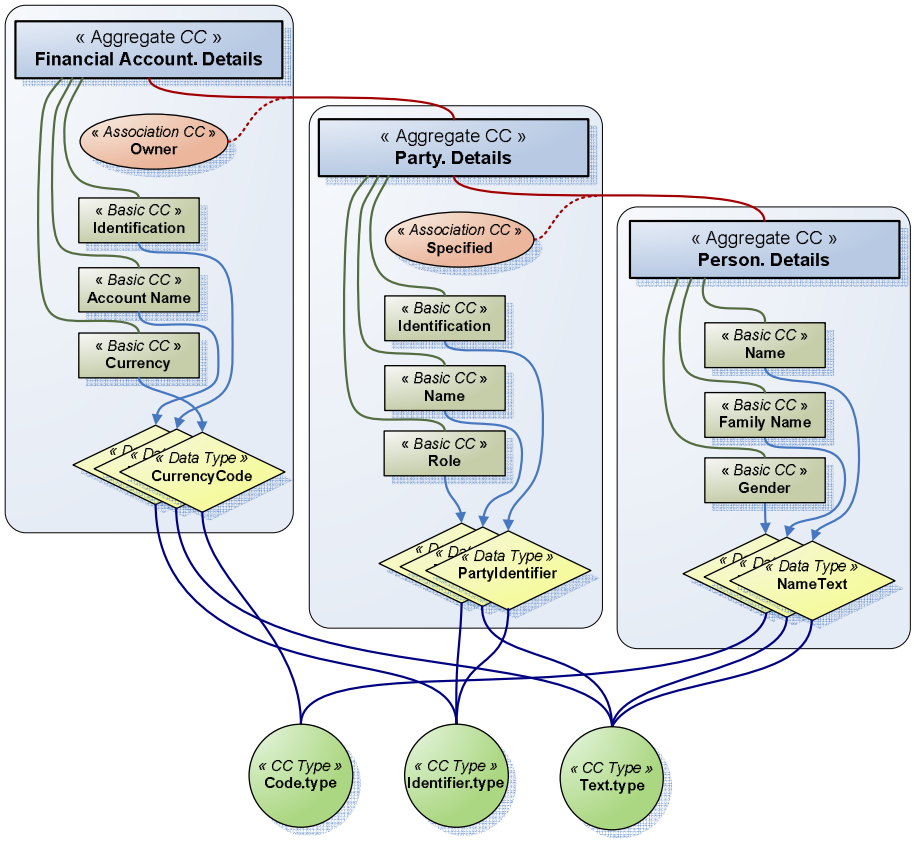

Figure 13 - Example (not exhaustive) of Financial Account Core Component with Data Types and CC Types

The key differentiator between CCs and Business Information Entities is the concept of Business Context. The formal description of a specific business circumstance is identified by the values of a set of Context Categories, allowing different business circumstances to be uniquely distinguished. When a CC is used in a real business circumstance it serves as the basis of a Business Information Entity. The Business Information Entity is the result of using a CC within a specific Business Context.

There are three different categories of Business Information Entities: Basic BIE, Association BIE, and Aggregate BIE. Figure 14 shows an example of a contextualised CC, i.e. a Business Information Entity.

It should be noted that, in this example, only the geopolitical context has specific values. No *RegistryObject* is created for any of the 7 other context categories since they each hold their default value.

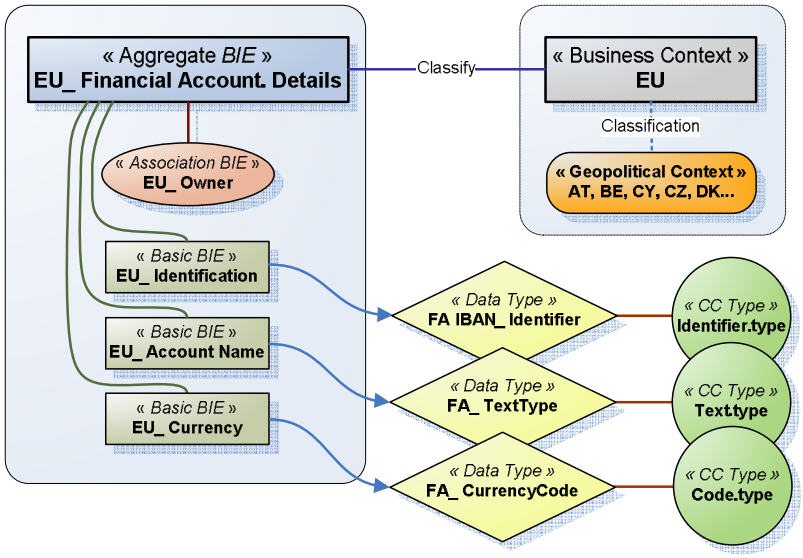

Figure 14 - Example (not exhaustive) of Financial Account Business Information Entity for a European Union context

CCs (and Business Information Entities) have Properties that are defined by Data Types. A Data Type represents the full range of values that SHALL be used for the representation of a particular CC Property.

The following subsections provide the UML class diagrams for [CCTS] Artefacts (BCC, ACC, ASCC…) that MUST be mapped to the RIM to ensure compliance with the [ebRIM].

3.2.1 Binding of the CC Information Model to the ebXML RIM

This section describes the UN/CEFACT Artefact for CC storage as outlined in Figure 8.

Figure 15 contains the UML model of the storage requirements for CCs. In the diagram *Registry* stored components are compliant with the [CCTS].

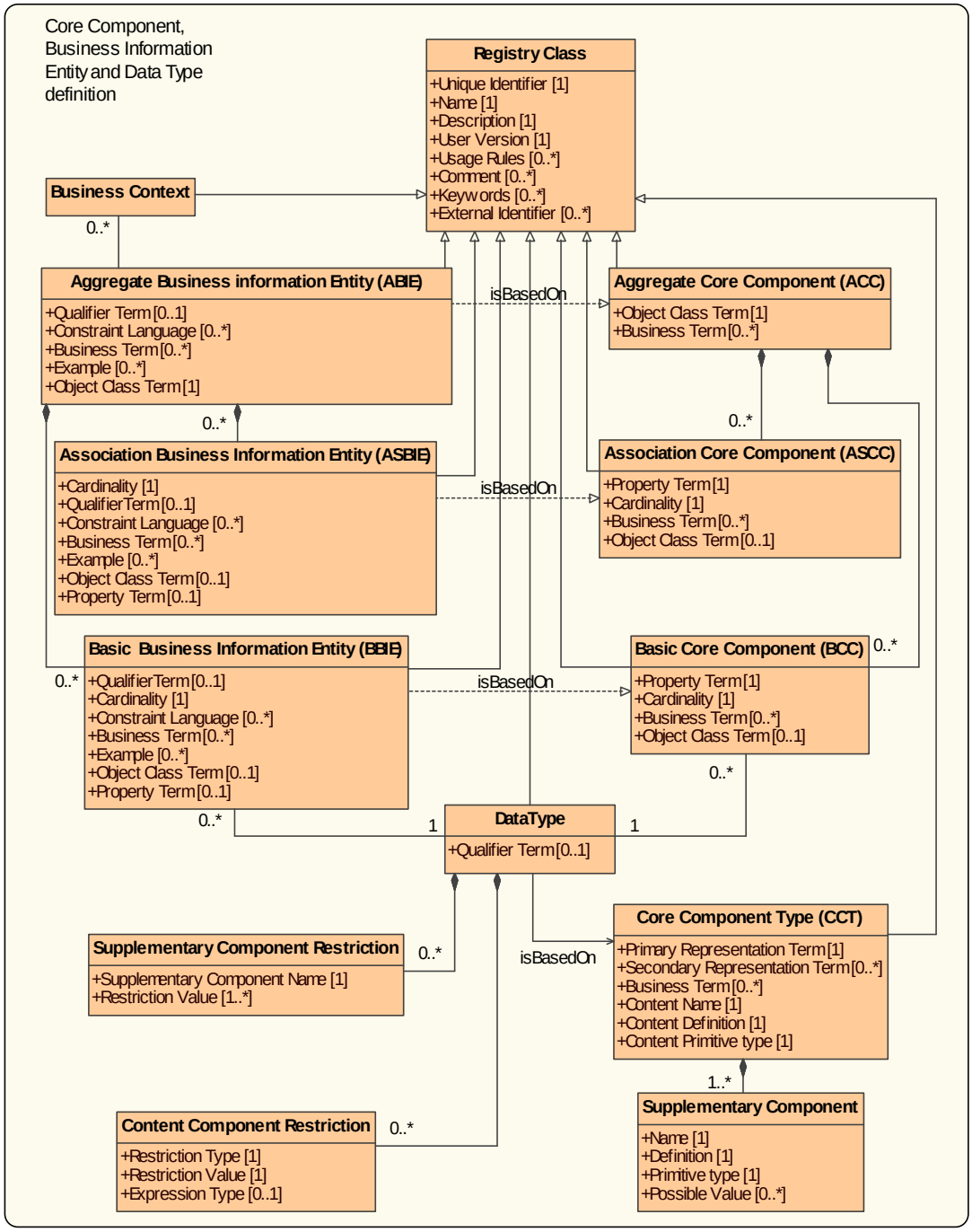

Figure 15 - Core Component *Registry* storage UML class diagram full definition

The Core Component Library that is maintained in the *Registry* contains the approved CCs, the submitted CCs, the deprecated CCs and the CCs that are being developed by registered users.

It should be noted that the root registry class attributes, illustrated in Section 3.1.2, Status, Start Date, User Version, Registrar, Registration Authority and Submitting Organization are naturally inherited by all child nodes of the parent components. This information SHALL be specified at the parent class level. The class Change History

SHALL only apply to the root registry class, unless explicitly identified otherwise by the submitter.

3.2.1.1 Binding of Core Component objects to the ebXML RIM

Table 10 lists the whole mapping profile for CC source model to [ebRIM] objectType.

Source Concept	[ebRIM] Object Type Name	[ebRIM] Parent Type Name	Comment
Core Component	Core Component	ExtrinsicObject	This item is used only for grouping all core component object types artefacts within the Registry Entry types classification tree. It is never instantiated directly for any business usage. Parent element is the canonical *ExtrinsicObject* Registry Entry type.
Registry Class	-	-	This corresponds to the Abstract *Registry* Class of the Global information storage view defined in Section 3.1. There is no corresponding Registry Entry; all its attributes and relationships are inherited directly by the sub-classes.
Aggregate Core Component	Aggregate Core Component	Core Component	
Association Core Component	Association Core Component	AssociationType	The Association Core Component establishes a new [ebRIM] association between the source ACC and the target ACC. This association is placed in the source ACC
Basic Core Component	Basic Core Component	Core Component	
Core Component Type	Core Component Type	Core Component	
Aggregate Business Information Entity	Aggregate Business Information Entity	Core Component	
Association Business Information Entity	Association Business Information Entity	AssociationType	ASBIE is mapped to registry association object and not considered as an extension of extrinsic object type. The Association Business Information Entity establishes a new [ebRIM] association between the source ABIE and the target ABIE. This association is placed in the source ABIE
Basic Business Information Entity	Basic Business Information Entity	Core Component	
Data Type	Data Type	Core Component	
Content Component Restriction	Content Component Restriction	Core Component	
Supplementary Component Restriction	Supplementary Component Restriction	Core Component	
Supplementary Component	Supplementary Component	Core Component	

Table 10 - Binding of Core Components to [ebRIM] objects and association types

Table 11 lists the whole binding of CC classes attributes to the RIM.

Source Attribute	[ebRIM] Attribute	Card.	Comment
Source concept: **Aggregate Core Component** (Sub-Node ExtrinsicObject/CoreComponent/ACC)			
Object Class Term	Slot (objectClassTerm, String, any string)	1..1	
Business Term	Slot (businessTerm, String, any string)	0..*	
Source concept: **Association Core Component** (Sub-Node of AssociationType)			
Object Class Term	Slot (objectClassTerm, String, any string)	1..1	
Business Term	Slot (businessTerm, String, any string)	0..*	
Property Term	Slot (propertyTerm, String, any string)	1..1	
Cardinality	Slot (cardinality, String, ["0..1" | "0..*" | "1..1" | "1..*"])	1..1	
Source concept: **Basic Core Component** (Sub-Node ExtrinsicObject/CoreComponent/BCC)			
Object Class Term	Slot (objectClassTerm, String, any string)	1..1	
Business Term	Slot (businessTerm, String, any string)	0..*	
Property Term	Slot (propertyTerm, String, any string)	1..1	
Cardinality	Slot (cardinality, String, ["0..1" | "0..*" | "1..1" | "1..*"])	1..1	
Source concept: **Aggregate Business Information Entity** (Sub-Node ExtrinsicObject/CoreComponent/ABIE)			
Object Class Term	Slot (objectClassTerm, String, any string)	1..1	
Property Term	Slot (propertyTerm, String, any string)	0..1	
Qualifier Term	Slot (qualifierTerm, String, any string)	0..1	
Business Term	Slot (businessTerm, String, any string)	0..*	
Constraint Language	Slot (constraintLanguage, String, any string)	0..*	
Example	Slot (example, String, any string)	0..*	
Source concept: **Association Business Information Entity** (Sub-Node of AssociationType)			
Object Class Term	Slot (objectClassTerm, String, any string)	1..1	
Property Term	Slot (propertyTerm, String, any string)	0..1	
Qualifier Term	Slot (qualifierTerm, String, any string)*	0..1	
Business Term	Slot (businessTerm, String, any string)*	0..*	
Cardinality	Slot (cardinality, String, ["0..1" | "0..*" | "1..1" | "1..*"])*	1..1	

Source Attribute	[ebRIM] Attribute	Card.	Comment
Constraint Language	Slot (constraintLanguage, String, any string)*	0..*	
Example	Slot (example, String, any string)*	0..*	
Source concept: **Basic Business Information Entity** (Sub-Node ExtrinsicObject/CoreComponent/BBIE)			
Object Class Term	Slot (objectClassTerm, String, any string)	1..1	
Qualifier Term	Slot (qualifierTerm, String, any string)*	0..1	
Business Term	Slot (businessTerm, String, any string)*	0..*	
Cardinality	Slot (cardinality, String, ["0..1" | "0..*" | "1..1" | "1..*"])*	1..1	
Constraint Language	Slot (constraintLanguage, String, any string)*	0..*	
Example	Slot (example, String, any string)*	0..*	
Source concept: **Data Type** (Sub-Node ExtrinsicObject/CoreComponent/DT)			
Qualifier Term	Slot (qualifierTerm, String, any string)	0..1	
Source concept: **Supplementary Component Restriction** (Sub-Node ExtrinsicObject/CoreComponent/SCR)			
Supplementary Component Name	Name	1..1	
Restriction Value	Slot (restrictionValue, String, any stgring)	1..*	
Source concept: **Component Component Restriction** (Sub-Node ExtrinsicObject/CoreComponent/CCR)			
Restriction Type	Name	1..1	
Restriction Value	Description	1..1	
Expression Type	Slot (expressionType, String, any string)	0..1	
Source concept: **Core Component Type** (Sub-Node ExtrinsicObject/CoreComponent/CCT)			
Primary Representation Term	Slot (primaryRepresentationTerm, String, any string)	1..1	
Secondary Representation Term	Slot (secondaryRepresentationTerm, String, any string)	0..*	
Business Term	Slot (businessTerm, String, any string)	0..*	
Content Name	Slot (contantName, String, any string)	1..1	
Content Definition	Slot (contentDefinition, String, any string)	1..1	
Content Primitive Type	Slot (contentPrimitiveType, String, any string)	1..1	
Source concept: **Supplementary Component** (Sub-Node ExtrinsicObject/CoreComponent/SC)			

Source Attribute	[ebRIM] Attribute	Card.	Comment
Name	Name	1..1	
Definition	Description	1..1	
Primitive Type	Slot (primitiveType, String, any string)	1..1	
Possible Value	Slot (possibleValue, String, any string)	0..*	

Table 11 - Binding of Core Component classes attributes to the ebXML RIM

3.2.1.2 Binding of Core Component associations to the ebXML RIM

This section defines the association type binding for CCs. It also provides the mapping from the source model associations to the RIM.

The Association Type column within Table 12 can be a canonical or a specific [ebRIM] ClassificationNode under the canonical AssociationType ClassificationScheme. Every instance of a [ebRIM] association object requires an associationType attribute that MUST be a reference to one of these ClassificationNodes.

The "Name" column corresponds to the name of the [ebRIM] association class Registry Entry instance, wich is not mandatory. The * in the [ebRIM] Association Type column indicates non canonical association types.

Table 12 lists the binding to the RIM of the CC associations as defined in Figure 15.

Association Source Object	Association Target Object	[ebRIM] Association Type	[ebRIM] Association Name
ACC	BCC	Contains	*${BCC Property Name}*
BCC	DataType	InstanceOf	HasDataType
ABIE	BBIE	Contains	*${BBIE Property Name}*
BBIE	DataType	InstanceOf	HasDataType
DataType	Supplementary Component Restriction	Contains	HasSupplementaryComponentRestriction
DataType	Content Component Restriction	Contains	HasContentComponentRestriction
DataType	CCT	InstanceOf	HasType
CCT	Supplementary Component	Contains	HasSupplementaryComponent
ABIE	ACC	RelatedTo	
BBIE	BCC	RelatedTo	
ABIE	Business Context	ContextFor*	
ACC	ACC	AssociationCoreComponent*	*${ASCC_Instance_name}*
ABIE	ABIE	AssociationBusinessInformationEntity*	*${ASBIE_Instance_name}*

Table 12 - binding to the ebXML RIM of the Core Components associations

Listing 4 shows an example of registry association for the BCC "Account.Name", a property for the ACC "Financial.Account".

```
<rim:Association id="${ASSOCIATION_IDENTIFIER_ID}"
      associationType="urn:oasis:names:tc:ebxml-
regrep:AssociationType:Contains"
      sourceObject="${SOURCE_Financial_Account_ACC_IDENTIFIER_ID}"
      targetObject="${TARGET_Account_Name_BCC_IDENTIFIER_ID}">
      <rim:Name>
            <rim:LocalizedString value="Account.Name"/>
      </rim:Name>
</rim:Association>
```

Listing 4 - Example of an [ebRIM] Association Class instance

Business Information Entities, the semantically qualified CCs, indirectly classified within the *Registry* by the association *ContextFor* with Business Context instances.

3.2.2 Multilingual management

This section describes how the multilingual capabilities of the *Registry* shall be managed.

CCs can be created in a given language. For UN/CEFACT standardisation requirements and for the internationalisation of components, submissions MUST be at a minimum in English (Oxford Dictionary) in order to comply with the [CCTS].

For an international use of the *Registry*, the multi language capabilities must be introduced.

The *Registry* storage and application MUST provide a mechanism where data and metadata can be easily accessed in other languages.

The following is the list of mandatory attributes for stored multilingual CCs:

- Name[8],
- Definition,
- Business Term,
- Object Class,
- Property Term,
- Representation Term,
- Qualifier Term,
- Comments.

Note: each of these mandatory attributes MUST be supplemented by an attribute which indicates the language of the attribute.

Acronyms and Keywords SHOULD be managed in multilingual format.

To identify the language, the language code list ISO 639–2 MUST be associated with the value of the attribute. (http://www.iso.org).

All this information MAY be used as search criteria. To find an Artefact in the *Registry*, it is possible to specify the language used in the criteria when a word is to be found in a definition, in an acronym, in a keyword or in a comment. By default, the language used for the search criteria is the current language used in the tools. This information is also used to see the detail of the definition of the component. By default, the language used to see the detail of the definition of the component is the current language used in the tools. If the definition in the current language is not found in the repository, the English definition is given.

3.2.3 Version management

This section describes the mechanisms to manage the evolution of the UN/CEFACT Artefacts contained in the *Registry* in order to ensure the integrity.

An important aspect of versioning is to ensure the complete management of the capabilities of the [CCTS] within the *Registry*. In the following two subsections, the versioning approach adopted by the [CCTS] is first described and then the corresponding mapping approach applied in the *Registry* is explained.

8 This corresponds to Dictionary Entry Name in the [CCTS]

3.2.3.1 Versioning in the [CCTS]

All objects managed in the [CCTS] (Registry Class) can change over time. The [CCTS] manages these changes by defining an attribute "Version" (mandatory) on different objects:

- On CCs:
  - Core Component Type,
  - Data type,
  - Basic CC,
  - Association CC,
  - Aggregate CC,
- On BIEs:
  - Data type,
  - Basic Business Information Entity,
  - Association Business Information Entity,
  - Aggregate Business Information Entity,
  - On Business Context.

There are also some constraints.

Contraints on the Classification Scheme: the number of Classification Schemes for Context Categories is currently established at eight. The *Registry* enables the number of classification schemes to be increased and the number of classification nodes within a classification scheme to be extended, as deemed necessary by owners of these schemes.

Contraints on the Context Value: the values for a Classification Scheme can not be modified; these values are initialised at the beginning and used in Business Context. So, only the Business Context can change. To evolve these values, it is possible to:

- Add Context Value, for a new value,
- Delete Context Value, if the value is not used in a Business Context.

Contraints on the CCT: all approved CCTs will be made available by the *Registry* implementation in accordance with designated release schedules.

The *Registry* includes support for metadata that is required to implement the following rules for version management of Registry Entries, specified in the [CCTS]:

- S66: Stored Registry Classes SHALL include a Version number to keep track of the evolution over time of a Registry Class,
- S70: Except for the first Version of a Registry Class, each stored Version SHALL be linked to its previous Version,

- S71: Except for the last Version of a Registry Class, each stored Version SHALL be linked to its next Version,
- S72: Stored Registry Classes SHALL include the history of the status lifecycle of each Version.

3.2.3.2 Mapping [CCTS] Version in the *Registry*

This section explains how Artefact versioning is managed in the *Registry*.

This paragraph lists the relevant attributes and classes needed to manage version in the *Registry* for CCs:

- The [CCTS] “Dictionary Entry Name” is mapped into the "Logical ID" attribute of the Registry Entry,
- The [CCTS] “Version” attribute is mapped into the “contentVersionInfo.comment” attribute of the *Registry* Entry Object,
- The Change History object is mapped to an Association in the *Registry*,
- The Previous version is mapped to the predefined association “Supersedes” in the *Registry*,
- The Next version is derived from “Supersedes” associations.

When a CC or a BIE is submitted, the complete Identification of the object is given: Dictionary Entry Name and Version.

To cater for general metadata storage rule S66 (Stored Registry Classes SHALL include a Version number to keep track of the evolution over time of a Registry Class) in the case where the Identification is incomplete (Version is omitted), the following steps will be applied:

- the Version attribute is set to 1 in the case of a new component,. An associated Change History object will be created with the following values:
    - Change type attribute is set to “new element”,
    - Change date attribute is set to the current date of the project,
    - Change Description attribute is set to “new element submitted”,
    - Request by attribute is set to the current requester,
    - Request Date attribute is set to the current date and time.
- the last version of the Registry Class in the *Registry* incremented by one will be used in the case of an existing Artefact. In addition it is necessary to find the latest version of the Artefact (identified by the logical ID in the *Registry*), and if the latest version of the Artefact exists in the repository, to create a link between the latest version found and the new submitted object. For this operation the following steps will be applied:
    - create the link “next version” from the Artefact found to the new Artefact,

- create the link “previous version” from the new Artefact to the Artefact found.

- A Change History object will be created in the *Registry* with the values:
  - Change type attribute is set to “new version”,
  - Change date attribute is set to the current date of the project,
  - Change Description attribute is set to “new version submitted”,
  - Request by attribute is set to the current requester,
  - Request Date attribute is set to the current date and time.

The [CCTS] Model forces the instantiation of objects unique in a specific version. The *Registry* MUST manage these associations between the objects.

To manage these associations, when a new version of an Artefact is submitted, the *Registry* automatically follows the rules specified in Table 13.

Change version of component below	ABIE	ASBIE	BBIE	ACC	BCC	ASCC	DT	BC
ABIE	create new version	create new version	create new version					
ASBIE	create new version	create new version	create new version					
BBIE	create new version	create new version	create new version					
ACC				create new version	create new version	create new version		
BCC				create new version	create new version	create new version		
ASCC				create new version	create new version	create new version		
DT								
BC								
CCT								

Table 13 - Object change version rules

The *Registry* MUST implement the following rules to manage the available associations:

- REP1: when an object is approved, the previous version is automatically passed to deprecated,
- REP2: when an object is submitted, the object CANNOT create a link to an object with the “deprecated” status.

3.3 Code set storage

Apart from the technical neutral data, the *Registry* can store other data. These other data are either "Business associated data" that are attached at any CC or BIE (e.g. legal information pieces) or operational components that derive from CC or BIE.

A CC or a BIE has no specific relationship to any given syntax. Through the Syntax Binding process a CC or a BIE can be expressed in any syntax, i.e. the technical neutral component becomes an operational component. The operational components have been transformed to formats such as XML, EDIFACT, etc. They are organised in Directories (e.g. UN/EDIFACT Directories, XML Schemas).

3.3.1 XML Schemas Directories

XML Schemas can derive from any (or set of) technically neutral component. These XML Schemas represent Directories that can be stored on the *Registry*. The user can search and retrieve them via the *Registry* Web site.

The UN/CEFACT Artefact for XML Schema Directory storage as outlined in Figure 8, is mapped to the RIM adopting the [ebRS] File Folder metaphor mechanism.

Each schema MAY be linked with the related CCs or BIEs if relevant or SHOULD be associated to a common Business Context instance.

3.3.2 UN/EDIFACT, UN/TDED, UN/LOCODE and related Directories

The *Registry* MUST maintain and provide access to the following Directories:

- UN/EDIFACT,
- UN/LOCODE,
- UN/TDED (ISO 7372),
- UN Recommendation code lists (e.g. Recommendation 20: Codes for Units of Measure used in International Trade),
- "ISO Country and Currency codes" Directory, by arrangement with ISO,
- Other relevant Directories.

Currently the UN/EDIFACT Directories and other related Directories are located on the UN/ECE Web site (http://www.unece.org/etrades/download/downindex.htm).

In order to streamline the management of all the United Nations e-business data and to improve the user service, these UN/EDIFACT Directories SHALL be stored in the *Registry*.

All these directories, as a special Artefact part of the Global information storage view (see Section 3.1) are mapped to the RIM adopting the [ebRS] File Folder metaphor mechanism.

Each data item MAY be linked with the related CCs or BIEs if relevant or SHOULD be associated to a common Business Context instance.

The end user will be able to search and retrieve data via a Web interface that supports the *Registry* search and retrieval services and protocols.

Any Directory Administrator will be able to manage the data that are under his responsibility: creation, modifications, deletion, and versioning.

3.3.3 Other Directories

If needed other Directories coded in other syntaxes can be stored in the *Registry*. These data can be searched and retrieved.

3.4 Reference document storage

3.4.1 *Registry* Package

A UN/CEFACT Project is managed by the *Registry* as a *Registry* Package. Figure 16 illustrates the typical content of a *Registry* Package which MAY include the XML metadata and attached documents that are referenced by the BRS and/or RSM, and the Technical Solutions that are derived from the RSM.

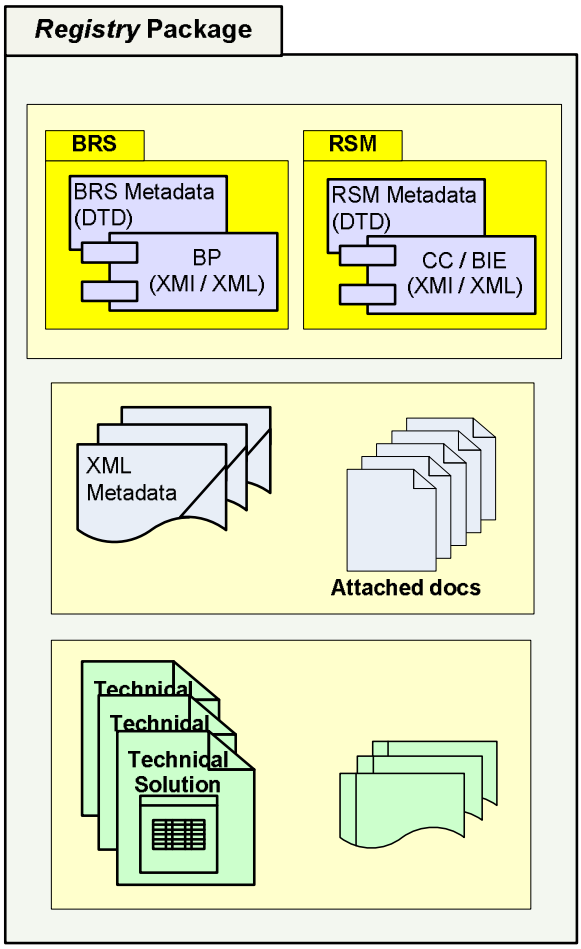

Figure 16 - *Registry* Package content

3.4.1.1 [ebRIM] profile for *Registry* Package

The Artefacts comprising a UN/CEFACT Project will be mapped to the RIM as a RegistryPackage object. The *Registry* Package object enables the logical grouping of *Registry Entries* associated with the project.

Every instance of a UN/CEFACT Project MUST correspond to a *Registry* Package instance. This object carries all the applicable administrative metadata (specified in Section 3.1) that is necessary to track the submission procedures and status of the project.

3.4.2 BRS Artefacts

This section describes the UN/CEFACT Artefact for BRS storage as outlined in Figure 8.

The BRS is developed within the TBG working group or project team in accordance with the UMM. The TBG project manager shall lead the project team in developing the BRS.

The BRS contains the necessary UMM artefacts (e.g. use case diagrams, collaboration diagrams, class diagrams, etc.). It is the formal document that describes the business requirements. The BRS MAY conform to the BRS DTD Template or it can be a text document

In this case the following metadata information SHALL be extracted from the BRS and submitted to the *Registry* in the appropriate XML exchange format (refer to Chapter 4). This metadata SHALL be compliant with the approved values of the classification schemes maintained in the *Registry*, as appropriate:

- Title
- Business context
  - Business domain
  - Business process(es)
  - Document identification
  - …
- Version
- Submitting organization.

This metadata provides key references to the associated set of documents. Collectively the BRS content becomes the foundation of the business information content of the *Registry*.

3.4.2.1 [ebRIM] profile for BRS

BRS documents are mapped to RIM as sub-node of the canonical *ExtrinsicObject* as shown in Table 14 and the document itself can be added to the *Registry* as repository item.

Source Attribute	[ebRIM] Attribute	Card.	Comment
Source concept: **Business Requirement Specification** (Sub-Node ExtrinsicObject/BRS)			
Unique Identifier	*Registry* Entry Unique Identifier	1..1	If no ID is provided by the submitter, the default system UUID registry identifier is used. [ebRIM] logical id attribute MAY have the same value.
External Identifier	ExternalIdentifier	0..*	
Document Title	Name	1..1	Canonical *Registry* Entry name attribute.
Objective	Description	1..1	Canonical *Registry* Entry description attribute.
Version	VersionComment	0..1	The *Registry* Entry attribute versionInfo is the official version for the. Any user versioning value used for other purposes can be added here. Same thing for repository item, if any. User version value can be mapped to ContentVersionComment attribute.
Usage Rule	Slot (usageRule, String, any string)	0..*	
Keywords	Slot (keywords, String, any string)	0..*	
Comment	Slot (comment, String, any string)	0..*	
Object Class Term	Slot (objectClassTerm, String, any string)	1..1	
Business Term	Slot (businessTerm, String, any string)	0..*	

Table 14 - [ebRIM] profile for BRS

3.4.3 RSM Artefacts

This section describes the UN/CEFACT Artefact for RSM storage as outlined in Figure 8.

Normally the BRS is accompanied by an RSM. The RSM MAY conform to the RSM DTD Template or it can be a text document

In this case the following metadata information has to be submitted to the *Registry* in the XML exchange format defined in the next chapter:

- Title
- Associated BRS
- Business context
  - Business domain
  - Business process(es)
  - …
- Version
- Responsible and submitting organization
- Revision history information.

It represents the technical content of the business documents of BPs defined in the BRS. It also documents any technology solution (such as XML or UN/EDIFACT) that is required.

The RSM requests one or more specific technology solutions for the BRS. It becomes the working document that is exchanged between the three operational groups for the development of the technology solutions.

The RSM originates in the TBG and is the key liaison document that is circulated between the operational groups during the BRS implementation process. The document goes through several steps before being finalised with an implementable technical solution.

All UMM artefacts defined in the document (CC, BIE, BC, etc.) MUST be added to the submission to the *Registry* in XMI format.

3.4.3.1 [eb RIM] profile for RSM

RSM documents are mapped to the RIM as sub-node of the canonical ExtrinsicObject as showed in Table 15 and the document itself can be added to the *Registry* as repository item.

Source Attribute	[ebRIM] Attribute	Card.	Comment
Source concept: **Requirement SpecificationMapping** (Sub-Node ExtrinsicObject/RSM)			
Unique Identifier	*Registry* Entry Unique Identifier	1..1	If no ID is provided by the submitter, the default system UUID registry identifier is used. [ebRIM] logical id attribute MAY have the same value.
External Identifier	ExternalIdentifier	0..*	
Document Title	Name	1..1	Canonical *Registry* Entry name attribute. The DEN could be used as a part of the Logical ID registry attribute.
Objective	Description	1..1	Canonical *Registry* Entry description attribute.
Version	VersionComment	0..1	The *Registry* Entry attribute versionInfo is the official version for the object. Any user versioning value used for other purposes can be added here. Same thing for repository item, if any. User version value can be mapped to ContentVersionComment attribute.
Usage Rule	Slot (usageRule, String, any string)	0..*	
Keywords	Slot (keywords, String, any string)	0..*	
Comment	Slot (comment, String, any string)	0..*	
Object Class Term	Slot (objectClassTerm, String, any string)	1..1	
Business Term	Slot (businessTerm, String, any string)	0..*	

Table 15 - [ebRIM] profile for RSM

3.5 Business process storage (not for prototype implementation)

This section describes the UN/CEFACT Artefact for BP storage as outlined in Figure 8.

BPs that are identified in the BRS will be described as UMM artefacts and presented as use case descriptions and diagrams. These BPs achieve the business objectives set out in the BRS and will process data specified as Message Structures and CCs.

The Common Business Process Catalog (CBPC) will be stored in the *Registry*. The information model to which common BPs MUST conform to is given in Figure 18 thereafter. It shows the objects and their relationships that specify the BP, and the various classification schemes that enable the CBPC to be searched by potential users.

These classifications together with the BP Names and Roles support the CCs Contexts as well as providing a Normative and Sub normative classification which will bring together any related BPs (based on the ISO classification and modified Porter Value Chain classification as described in the CBPC specification).

The CBPC information model is mapped to the RIM. This mapping is shown in Table 16.

The details of the Common BPs MAY be expressed in XML format following the BPSS.

3.5.1 Business Process Classification

3.5.1.1 Normative Category

Each BP in the CBPC is at its most general level represented by a *normative category* of inter-enterprise activity. The *normative categories* are one example of the division of an enterprise into *business areas* as recommended by the UMM[9].

None of the reviewed classification schemes directly satisfy all requirements, but the one that comes closest is the Porter Value Chain. The categorization selected for the catalogue is based on development and extension of that model.

The categories selected are example components for the Business Categories as reflected by the TBG Groups:

- Supply Chain,
- Transportation and Logistics,
- Customs,
- Finance,
- Architecture and Construction,
- Statistics Collection and Reporting,
- Insurance,

9 UMM, proposed Chapter 8, section dealing with the Business Domain View (BDV).

- Travel, Tourism and Leisure,
- Healthcare,
- Social Services,
- Accounting and Audit,
- Environmental,
- Agriculture.

They will have to be defined and maintained within a formalized classification schema.

3.5.1.2 Normative Sub-Category – The ISO Open-edi Phases of a Business Collaboration

Once an enterprise has been divided into thirteen high-level business areas, it is possible to decompose each one of those areas further into process areas.

The CBPC's second-level classification – its normative sub-categories – is based on the five successive phases of business collaboration as defined by the ISO Open-edi model. These phases are illustrated and explained in Figure 17:

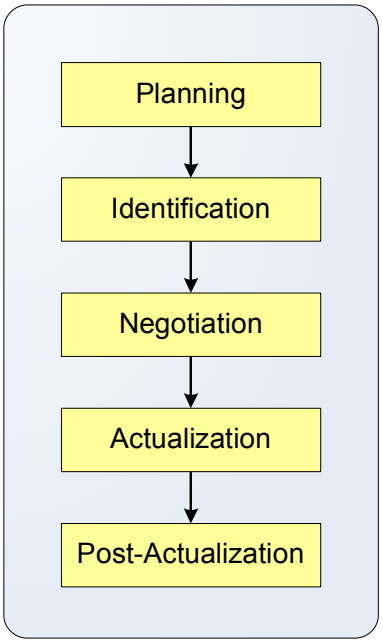

Figure 17 - Phases of a Business Collaboration[10]

3.5.2 Binding to the ebXML RIM of the BP Classification

The normative categories and sub-categories are mapped to the RIM as sub-nodes of the BP Context defined in Section 3.1.1. Listing 5 shows a part of the normative category mapping to the RIM.

10 from the ISO Open-edi Reference Model

```
<!-- Defining Business Process Context Classification Scheme with a sub-set
of the normative category list -->
<ClassificationScheme
lid="urn:un:unece:uncefact:icg:registry:ContextClassification:BusinessProces
s"
id="urn:un:unece:uncefact:icg:registry:ContextClassification:BusinessProcess
" isInternal="true" nodeType="urn:oasis:names:tc:ebxml-
regrep:NodeType:UniqueCode">
      <Name>
            <LocalizedString charset="UTF-8" value="Business Process
Context" xml:lang="en-US"/>
      </Name>
      <ClassificationNode code="NormativeCategory"
parent="urn:un:unece:uncefact:icg:registry:ContextClassification:BusinessPro
cess"
lid="urn:un:unece:uncefact:icg:registry:ContextClassification:BusinessProces
s:NormativeCategory">
            <ClassificationNode code="SupplyChain"
parent="urn:un:unece:uncefact:icg:registry:ContextClassification:BusinessPro
cess:NormativeCategory" lid="NormativeCategory:SupplyChain">
                  <ClassificationNode code="SupplyChain:Planning"
lid="SupplyChain:Planning"/>
                  <ClassificationNode code="SupplyChain:Identification"
lid="SupplyChain:Identification"/>
                  <ClassificationNode code="SupplyChain:Negotiation"
lid="SupplyChain:Negotiation"/>
                  <ClassificationNode code="SupplyChain:Actualisation"
lid="SupplyChain:Actualisation"/>
                  <ClassificationNode code="SupplyChain:Post-Actualisation"
lid="SupplyChain:Post-Actualisation"/>
            </ClassificationNode>
            <ClassificationNode code="TransportationLogistics"
parent="urn:un:unece:uncefact:icg:registry:ContextClassification:BusinessPro
cess:NormativeCategory" lid="NormativeCategory:TransportationLogistics">
                  <ClassificationNode
code="TransportationLogistics:Planning"
lid="TransportationLogistics:Planning"/>
                  <ClassificationNode
code="TransportationLogistics:Identification"
lid="TransportationLogistics:Identification"/>
                  <ClassificationNode
code="TransportationLogistics:Negotiation"
lid="TransportationLogistics:Negotiation"/>
                  <ClassificationNode
code="TransportationLogistics:Actualisation"
lid="TransportationLogistics:Actualisation"/>
                  <ClassificationNode code="TransportationLogistics:Post-
Actualisation" lid="TransportationLogistics:Post-Actualisation"/>
            </ClassificationNode>
      </ClassificationNode>
</ClassificationScheme>
```

Listing 5 - Mapping to [ebRIM] Classification of the Normative Category (not exaustive)

3.5.3 Business Process Information Model

The CBPC information model is an abstract data structure defining the content and structure of the catalog.

First, we provide the semantics of each class, and then we focus on the semantics of relationships in the information model. Figure 18 shows this information model.

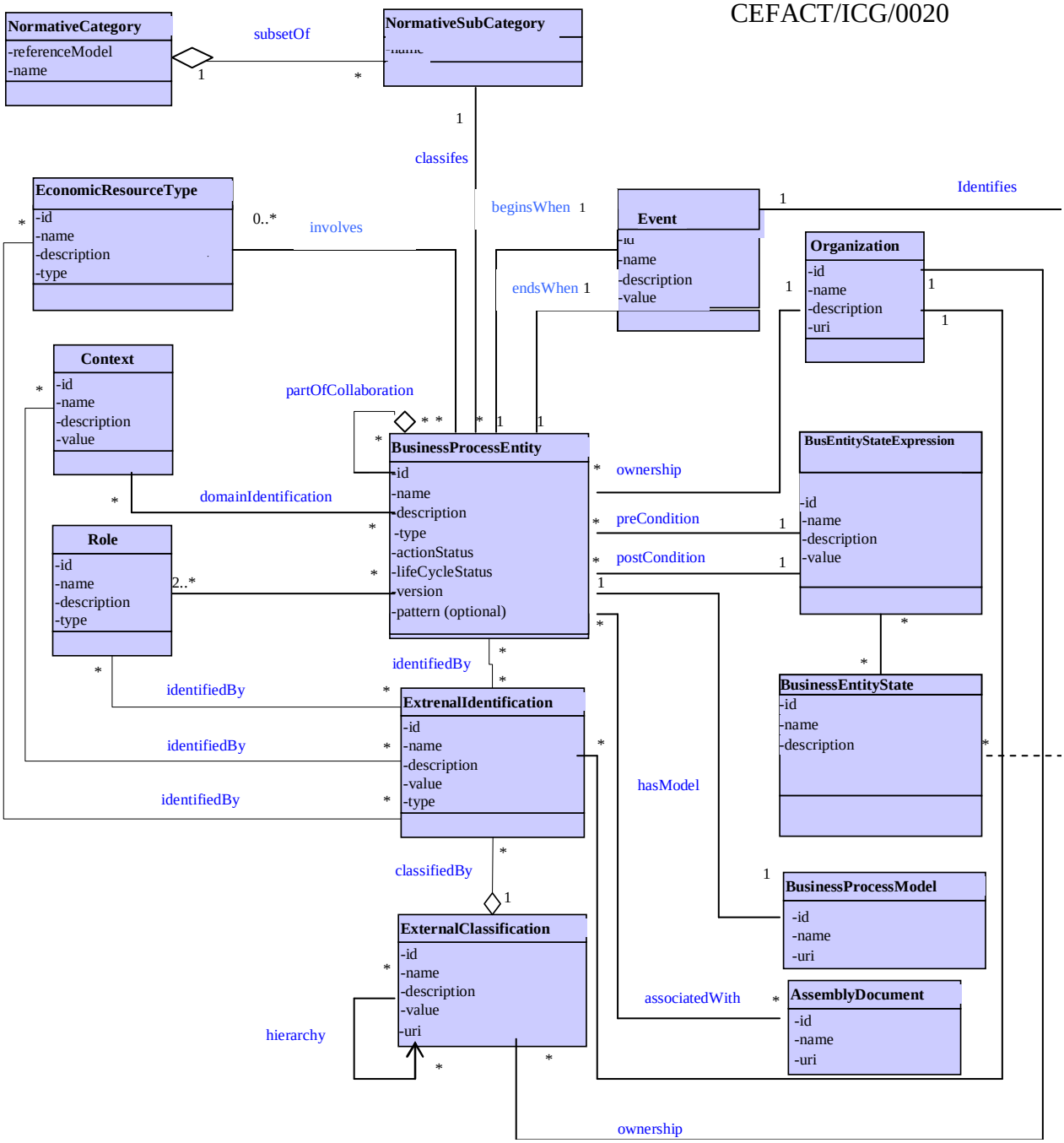

Figure 18 - Abstract Data Structure for the CBPC Information Model

3.5.4 [ebRIM] profile for CBPC

CBPC metadata	ebXML RIM equivalent	Comment
Business Process Entity	Extrinsic Object	It MAY be either transaction or collaboration
Business Process Model	Slot	Business Process model will be added as slot of Extrinsic Object
Assembly Document	Slot	Assembly document will be added as slot of Extrinsic Object
Normative Category	Classification Scheme	Normative category values will appear as child node under UN-CEFACT classification scheme in tree
Sub normative category	Classification Node	Sub Normative category values will appear as child node under Normative category values
User	User	
Organization	Organization	
Metadata		
Economic Resource Type	Classification Node	Child node under CBPC classification scheme in tree and values of economic resource type will appear as child of this classification node and can be associated with catalog being submitted.
Context	Classification Node	Will appear as child scheme node under CBPC classification scheme in tree and values of context type will appear as child of this classification node and can be associated with catalog being submitted.
Role	Classification Node	This will appear as classification scheme node under CBPC classification scheme in tree and values of Role type will appear as child of this classification node and can be associated with catalog being submitted.
Business Entity State	Classification Node	This will appear as classification scheme node under CBPC classification scheme in tree and values of Business Entity State will appear as child of this classification node and can be associated with catalog being submitted.
Event	Classification Node	This will appear as classification scheme node under CBPC classification scheme in tree and values of Event will appear as child of this classification node and can be associated with catalog being submitted.

CBPC metadata	ebXML RIM equivalent	Comment
Relationship		
Involves	Classification Node	All the relationships MUST appear as child nodes of the CBPC Association type classification scheme.
Part of collaboration	Classification Node	-do-
Begin When	Classification Node	-do-
Ends When	Classification Node	-do-
Pre condition	Classification Node	-do-
Post Condition	Classification Node	-do-
Involved party	Classification Node	-do-
Identified by	Classification Node	-do-

Table 16 - CBPC information model mapping to the ebXML RIM

4 Data exchange format (input/output)

This section defines the transformations that are required in order to provide automatic submission and retrieval process of *Registry* Artefacts.

The most important task of such transformations is to enable the preservation of all the required information in the target ebXML RIM format. In order to ensure this, the targeted RIM format SHALL be able to store all the data defined in every recognised input and output exchange format. In addition it SHALL be able to maintain their structure.

The system SHALL permit an incremental completion of the information about stored Artefacts of the corresponding Registry Entries. The [CCTS], for example, defines a multitude of data and metadata for its various objects. It SHOULD be possible to submit a complex component, such as an ABIE, in an incremental way thus permitting the system to be able to create new Registry Entries without all the necessary information that is required to completely define the object in question within the *Registry*. The missing information could be provided through other mechanisms (by adopting default values or inheritance rules where possible for example).

For the submission and retrieval of Artefacts the *Registry* MUST use XML structures as the basis for data exchange and SHALL provide an XML generic structure for every UN/CEFACT Artefact. These XML structures are called the *Registry* exchange format.

4.1 Registry exchange format

The *Registry* exchange format is a set of XML schemas that enable the exchange of UN/CEFACT Artefacts. Such an exchange format facilitates structure and type validation.

The exchange format allows the verification of the structure and the types of submitted Artefacts. It is a generic format that is also a key to registry federation. On the receipt of a *Registry* submission, the constituent Artefacts, once validated, MUST be stored in the ebXML registry. The API SHALL be responsible for transforming the exchange format into the RIM format. The API is responsible for registry conversion (RIM ←→ exchange format) and interrogation (get list of ABIE, get complete ABIE with BBIE).

4.2 BRS and RSM exchange format

In order to streamline the system management each BRS and RSM syntax neutral piece of information will be provided in a UN/CEFACT profiled XMI document. In the same way each syntax neutral stored piece of information MAY be retrieved in the same profiled XMI format.

Both the BRS and RSM SHALL respect a compliant XML DTD structure and will be stored respecting these structures (Word/Open Office patterns). It will then be possible to create web forms that correspond to the BRS and RSM structures which will enable the simplification of data acquisition.

However, during a transitional period until the DTD structures have been formally approved, other formats (spreadsheet, PDF, etc.) will be accepted.

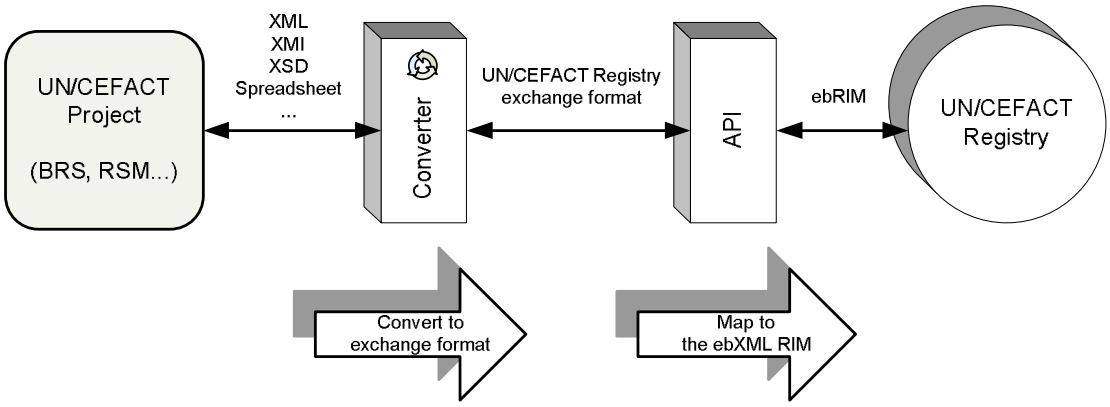

Figure 19 - Data exchange principle

4.3 Registry input flow

The diagram in Figure 19 represents the processing steps required to submit and catalogue a new BRS and RSM that has been submitted by an organization to UN/CEFACT.

Only registered users will be able to submit Artefacts for approval and subsequent publication through the *Registry*. Artefacts not in the *Registry* exchange format will need to be converted into the *Registry* exchange format by a client application (i.e. the

Converter as in Figure 19). Following submission to the *Registry* all Artefacts are automatically validated by the content management service to ensure compliance with the relevant UN/CEFACT standards and procedures (for example, a BIE can be created only if a CC already exists).

4.4 CC exchange format

The *Registry* exchange format enables the simplification of the submission of CCs through a readable and [CCTS] compliant format. The exchange format is based on the UML classes of the [CCTS] in order to ensure data integrity and consistency (structure and type). Other exchange formats such as XMI or XML Schema instances will be used to represent instance CC structures.

The exchange format defines all the information required and thus enables the CC Structure and Meta Data to be preserved. It is worth noting that the XMI format is a standardised exchange format for UML tools. The TMG is currently developing UML profiles for UN/CEFACT Artefacts (i.e. a UMM profile for UML and a CCTS profile for UMM).

In order to implement the *Registry* exchange format for a compliant design of the CCTS, it will be important to initially create an XML Schema profile for the UML Class diagram. However, all [CCTS] rules cannot be validated merely through the exchange format. For example, it is not possible to ensure that the component name will be correct (content, [CCTS] naming rules). Such rules consequently will have to be integrated into the API or, failing that, will have to be directly controlled manually by UN/CEFACT groups during the approval process.

4.5 CCTS UML mapping to the ebXML RIM

The CCTS UML mapping to the RIM permits the generation of the RIM structure enabling it to be stored in an ebXML registy. An ebXML registry is a standard that ensures the compatibility of architectural structures between registries to be maintained. This will be very important for the federation of registries. The mapping of a UML class to the RIM requires the mapping rules of all UML components used in the [CCTS] (class, association) and their corresponding RIM components to be described.

4.6 Other exchange formats

4.6.1 UML/XMI

All UML diagrams SHALL be provided as a deliverable from the *Registry* in the UN/CEFACT Profiled exchange format XMI which is the standard exchange format for UML. UN/CEFACT is currently developing an XMI profile to ensure consistent use of the XMI language, thus enabling consistent input and output of the information for use by standard UML development tools.

The profile is currently being refined by the TMG and once stabilised its structure will be validated prior to its insertion into the *Registry*.

4.6.2 Spreadsheet

The TBG Harmonization group has defined a spreadsheet format in order to facilitate the management of CC information within the groups remit. This spreadsheet corresponds to a set of structured data and can be transformed automatically into the standard UN/CEFACT exchange format in order to facilitate submission, validation and storage of newly defined CCs in the *Registry*.

4.6.3 NDR-XSD

The [NDR] defines an XML Schema for exporting CCTS (BIE and CC). This format enables the creation and validation of an ebXML message based on CCs.

5 Federation (not for prototype implementation)

5.1 ebXML registry federation principle

The [ebRS] defines registry federation as a group of registries that have voluntarily agreed to form a loosely coupled union. Such a federation may be based on common business interests and specialties that the registries share. Registry federation appears as a single logical registry to registry clients.

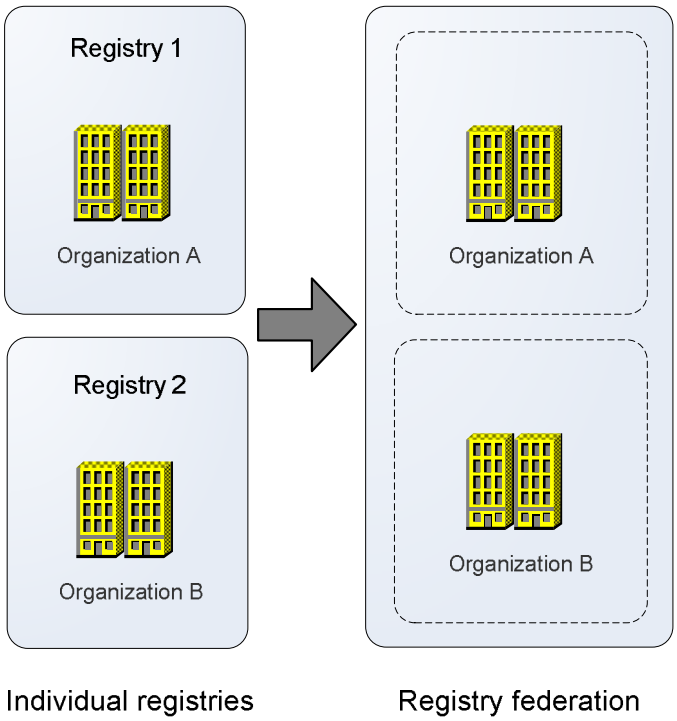

Figure 20 - Registry federation[11]

11 From [ebRS]

Registry federations are based on a peer-to-peer (P2P) model where all participating registries are equal. Each participating registry is called a registry peer. There is no distinction between the registry operator that created a federation and those registry operators that joined that Federation later. Any registry operator may form a registry federation at any time. When a federation is created it must have exactly one registry peer which is the registry operated by the registry operator that created the federation.

5.2 UN/CEFACT Registry federation

The *Registry* SHALL support the federation with other business registries based on the same architecture. The ICG on behalf of UN/CEFACT will act as registry Operator for the *Registry* federation as defined in [ebRS]. Participants to the federation MUST be validated by the *Registry* Operator, MUST support the *Registry* information model and MUST conform to the *Registry* Charter.

The federation has to be able to share all UN/CEFACT Artefacts and objects (e.g. ABIE, ACC, Business Context, BP).

The federation must ensure the capability to declare related objects stored in different registries (e.g. a BIE can be based on an ACC and the two objects can be stored in different registries).

Figure 21 provides an example of registry federation architecture where several national registries as well as registries representing business sectors are federated to the *Registry*.

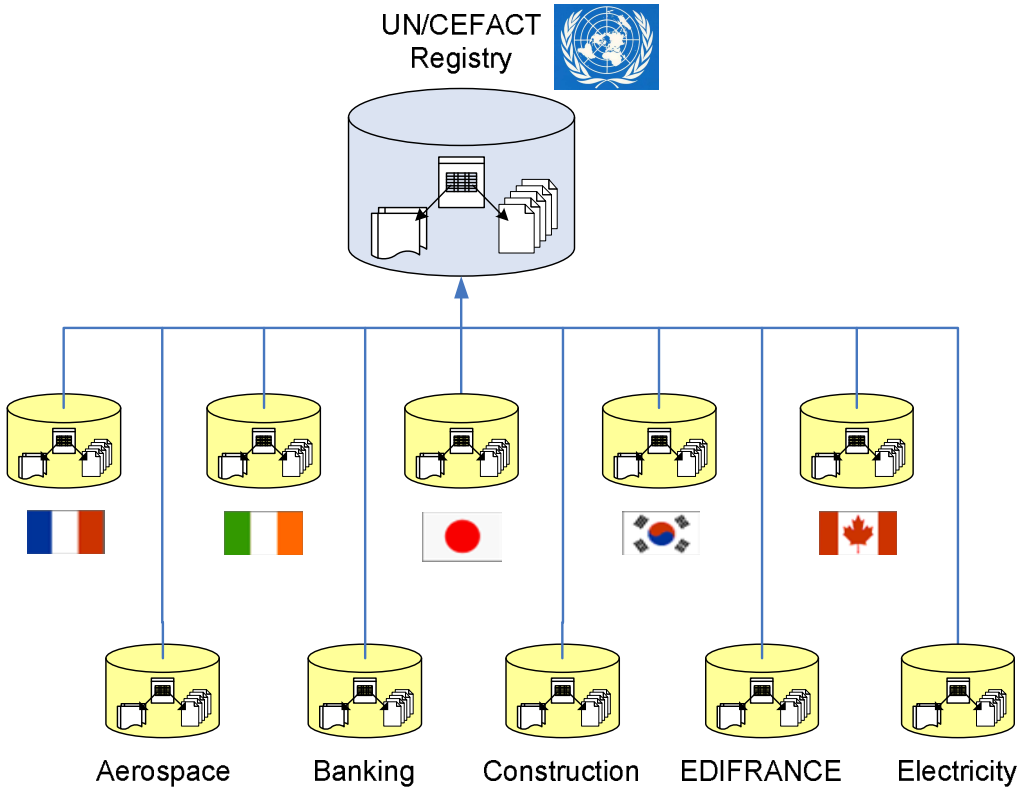

Figure 21 - UN/CEFACT *Registry* federation architecture (example)

Based on this example, let's take a look at the dynamics behind the *Registry* federation. In Figure 22 a user makes a query to the *Registry* (step 1). This query is then broadcast to all the federated registries (step 2) which in turn may broadcast the query to their other registries. In the diagram, the EDIFRANCE registry broadcasts the query to the registries which are independently federated to it (step 2-a). Finally all queried registries provide any relevant replies to their parent registry (step 2-b and step 3). The *Registry* provides the reply to the query to the user (step 4).

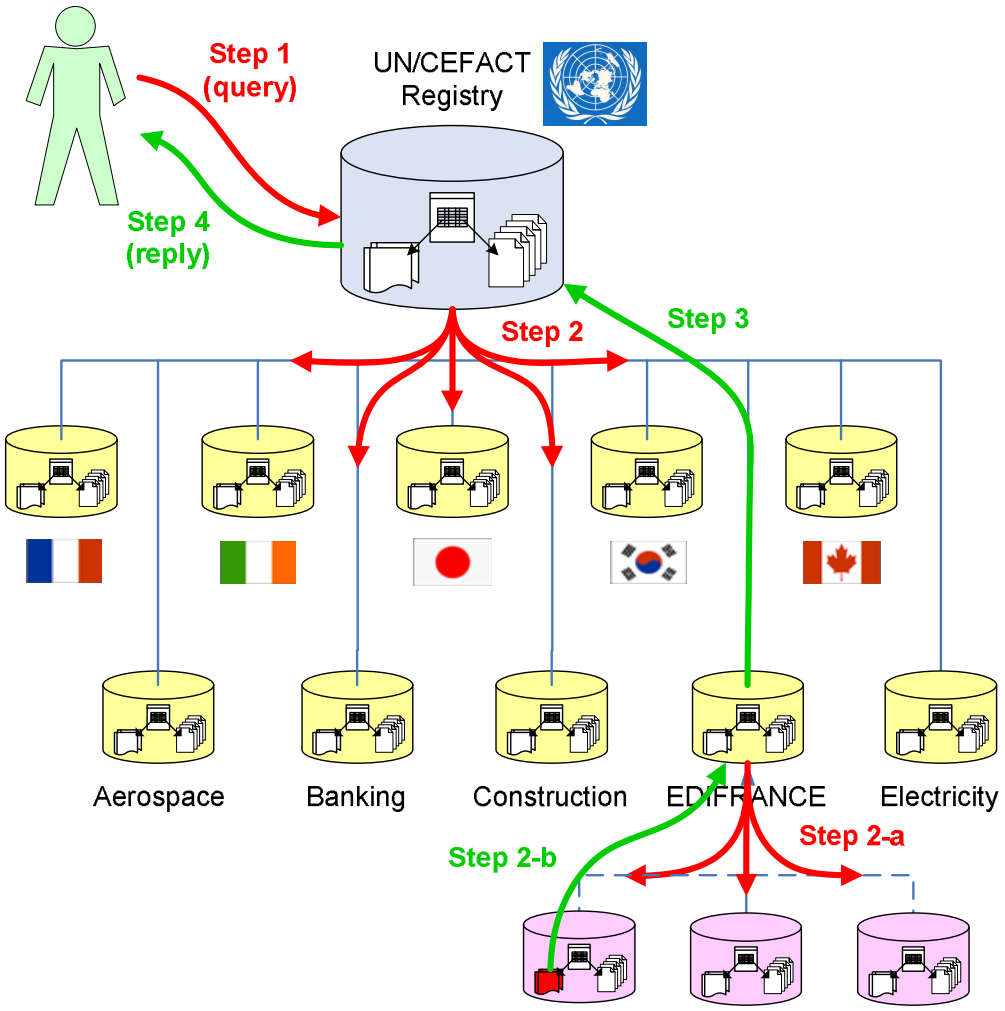

Figure 22 - example of a federated query to the *Registry*

NB: in the diagram in Figure 22 the dotted arrows represent the associations between the EDIFRANCE registry and all its federated registries, which are not federated to the UN/CEFACT Registry.

6 Conformity tests (not for prototype implementation)

6.1 Conformance

The *Registry* SHALL conform to the [ebRIM] in its entirety and to specific parts of the [ebRS]. This section identifies all specific features of the [ebRS] as well as all specific sections of this specification to which the *Registry* MUST conform to.

Conformance also applies to other registries that are federated to the *Registry*. Whether these registries fully implement the mandatory features is essential to having a robust, interoperable federation of registries around the *Registry*. In addition to general mandatory features for ebXML registry conformance as mentioned above, there exist some specific features that are mandatory for the *Registry* federation and that are also described in this chapter.

Finally this chapter provides guidelines to help develop conformance test suites and tools.

6.2 Mandatory features

NB: to be completed in a later version of this specification.

6.3 Test suite

NB: to be completed in a later version of this specification.

6.4 Procedures for testing

NB: to be completed in a later version of this specification.

6.5 Qualified body to do testing

NB: to be completed in a later version of this specification.

7 Registry services profile

The *Registry* serves as the storage and retrieval facility for all UN/CEFACT Artefacts.

Several ebXML registry services, illustrated in Figure 23, can be adapted to UN/CEFACT needs.

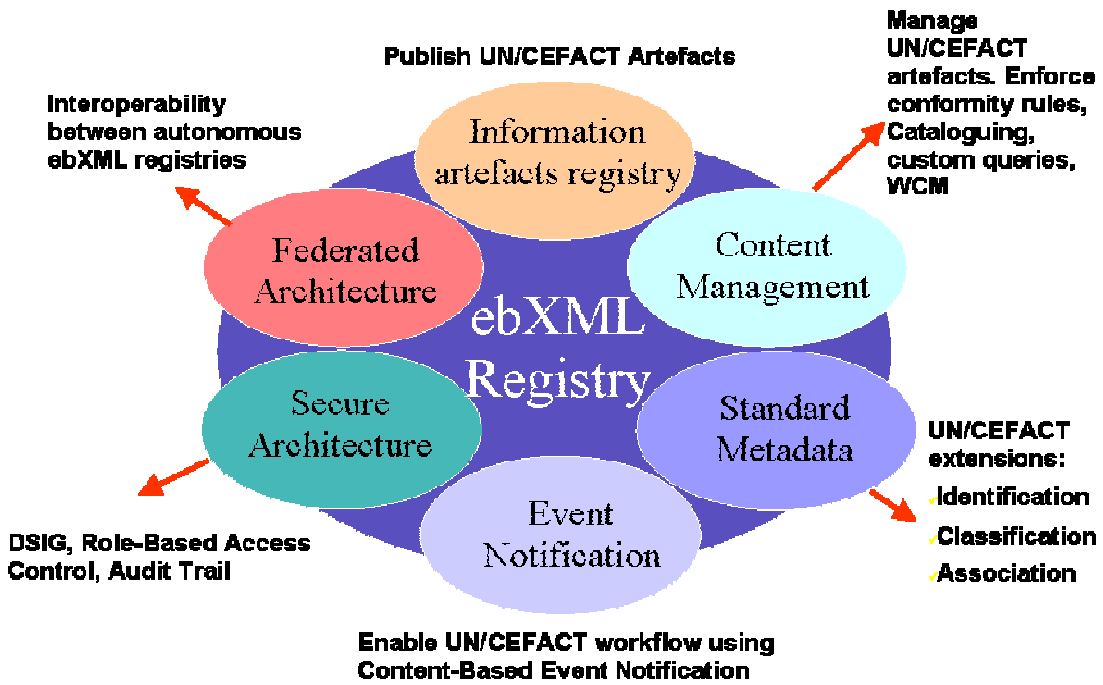

Figure 23 - ebXML Registry services features[12]

7.1 Content Management Service Profile

7.1.1 Content Validation Services

A set of CCs can be submitted in different formats. Common exchange formats are: spreadsheets, XML schema instances (complaint with the UN/CEFACT NDR document), XMI and other specific XML formats.

These formats typically are not capable of carrying all the metadata and content described by the [CCTS], but they can be considered as a submission format under the condition that they have and respect a minimal set of information and rules to validate CCs publishing.

Content validation mechanism MUST verify that [CCTS] rules are respected. For example mandatory attributes MUST be present (as *RegistryObject* name, description, Object Class Term…), naming rules SHALL be respected (the approved CC's name MUST be unique, the structure MUST be correct…).

7.1.2 Content Cataloging Services

An ebXML registry provides the ability for a user defined content cataloguing service to be configured for each ObjectType defined by the mapping. The purpose of the cataloguing service is to selectively convert content into [ebRIM] compatible metadata when the content is submitted. The generated metadata enables the selected content to be used as parameter(s) in a domain specific parameterized query.

12 Adapted from OASIS ebXML Registry Technical Committee work

7.2 Discovery Queries Profile

The *Registry* provides the ability to create predefined search and discovery queries as instances of the [ebRIM] *AdhocQuery* class. These predefined queries are created and/or managed by the *Registry* administrator upon requests from registered users.

***Registry* Artefact**	**Parameters**	**Description**
Core Component Artefacts		
Business Context	$ID $Name substring($Definition) $Version $Status one or more context classification $value Linked BIE $ID linked BIE $Name	Find Business Context instances by specifying values for one or more of these parameters.
ACC	$ID $Name substring($Definition) $Business term $Property term $Object Class Term $Version $Status $Owner $submission date $validation date	Find Aggregate Core Component instances by specifying values for one or more of these parameters.
ASCC	$ID $Name substring($Definition) $Business terms $Property term $Object class term Source ACC $ID Source ACC $Name Target ACC $ID Traget ACC $Name $Version $Status Owner $ID Owner $Name $submission date $validation date	Find Association Core Component instances by specifying values for one or more of these parameters
BCC	$ID $Name substring($Definition) $Business term $Property term $Object Class Ter Core Component Type $ID Core Component Type $Name Core Component Type $Primary representation term Data Type $ID Data Type $Name $Version $Status Owner $ID Owner $Name	Find Basic Core Component instances by specifying values for one or more of these parameters.
ABIE	$ID $Name substring($Definition) $Business term $Property term	Find Aggregate Business Information Entity instances by specifying values for one or more of these parameters

	$Object class term $Version $Status Owner $ID Owner $Name $submission date $validation date Business Context $value/s	
ASBIE	$ID $Name substring($Definition) $Business terms $Property term $Object class term Source ACC $ID Source ACC $Name Target ACC $ID Traget ACC $Name $Version $Status Owner $ID Owner $Name $submission date $validation date Business Context $value/s	Find Association Business Information Entity instances by specifying values for one or more of these parameters
BBIE	$ID $Name substring($Definition) Business Context $value/s $Business term $Property term $Object Class Term Core Component Type $ID Core Component Type $Name Core Component Type $Primary representation term Data Type $ID Data Type $Name $Version $Status Owner $ID Owner $Name	Find Basic Business Information Entity instances by specifying values for one or more of these parameters
DT	$ID $Name substring($Definition) Core Component Type $ID Core Component Type $Name Core Component Type $Primary representation term $Version $Status Owner $ID Owner $Name	Find Data Type instances by specifying values for one or more of these parameters
CCT	$ID $Name substring($Definition) $Primary representation term Data Type $ID Data Type $Name $Version $Status Owner $ID Owner $Name	Find Core Component Type instances by specifying values for one or more of these parameters

Table 17 - *Registry* discovery profile for Core Components

7.3 Event Notification Profile

An ebXML registry provides the ability for a user or an automated service to create a subscription to events that match a specified criterion. Whenever an event matching the specified criteria occurs, the registry notifies the subscriber that the event transpired.

7.3.1 Use Cases for Event Notification

The following are the most common use cases that MAY be implemented for the CC domain:

- an email notification MAY be sent when status changes are applied to aggregate components (ACC and ABIE);
- an email notification MAY be sent when new objects are submitted;
- an email notification MAY be sent when aggregate objects change;
- an alert MAY be created by users on updating of some interested components instances.

7.4 Security Profile

An ebXML registry provides a powerful and extensible access control feature that makes sure that users MAY only perform those actions on a RegistryObject or repository item for which they are authorized.

8 Application Interfaces

8.1 Specialized Interface

The specialized interface supports user and application interaction with UN/CEFACT Artefacts, e.g. through modelling tools, and provides methods to control the structure, semantic and syntax rules for submitted Artefacts. The Specialized interface interacts with the ebXML registry/repository through the JAXR (Java API for XML Registries) API.

A "Specialized Interface" SHALL provide access to privileged interrogation and submission functionality for stored components in registry and service Administration.

On each call, the Client Application SHOULD verify a user's role and right to access a defined function against the service's access policy (Section 9.1).

Users may connect to the *Registry* service through: 1) a web browser, 2) an external application (Thick client UI, dedicated software …) for "Retrieval" functions or 3) a "Workflow API" for "Submission" functions. A set of Web Services could be defined that can be called by various types of client applications. The envisaged Web Services could translate users' queries in *Registry* queries.

The *Registry* as an ebXML registry implementation comprises a set of interfaces designed to manage the objects and inquiries associated with the ebXML registry. The two primary interfaces for the *Registry* Service consist of:

- An interface that provides a collection of methods for managing objects within the *Registry*,
- An interface that controls the discovery and retrieval of information from the *Registry*,
- An interface that provides a collection of methods (such as "validate data", "retrieve documents", "initiate project", "transform format"…) for managing specific UN/CEFACT objects.

A client program utilizes the services of the *Registry* by invoking methods on one of the above interfaces defined.

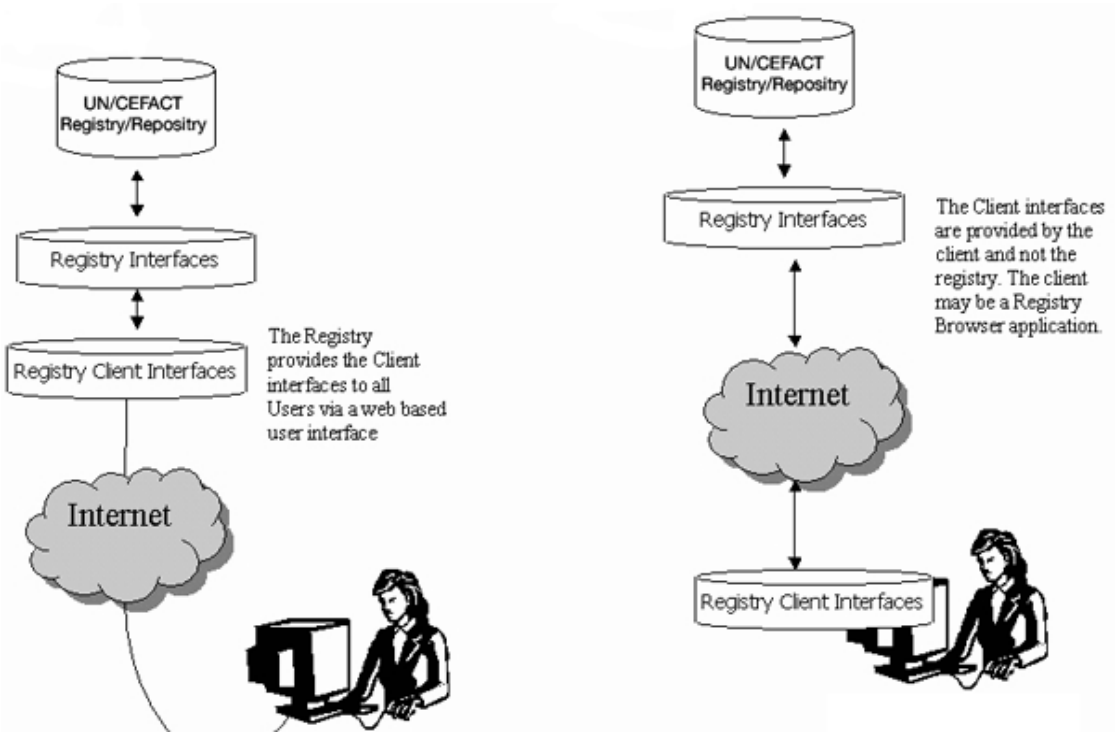

Figure 24 - *Registry* client interface options [13]

8.1.1 Submission Interface

Access to "Submission" functions is restricted to registered users with a "submitter role".

"Submission" function may only be accessed via the "Worflow API", direct http/soap access is not allowed to client application.

A submitter of components will be considered to be the components' owner.

13 Figure from [ebRS]

When submitting a new component or a new set of components, Client Application Interface Web Services receive the components description in XML format data input as described in Section 4.4 (CC exchange format).

The submission follows the process described below:

- Authenticate user,
- Check input document compliance :
  - check document structure (return missing mandatory elements),
  - check document content compliance : naming rules, linked components status, versioning…
- Translate document from input format to the RIM format,
- Add data into the *Registry*,
- Generate IDs for new objects,
- Fill input document with new IDs and return it to caller.
- Mandatory functions:
- Add, delete or update components. (e.g. update ABIE with ID…),
- Add, delete or update Business Specification,
- Add, delete or update Documents associated with Business Specification,
- Submit a new version.

8.1.2 Retrieval Interface

8.1.2.1 Search functions

Access to "Search functions" is not restricted; *RegistryObject* visibility is governed by the user's role and *Registry* Artefact access policies.

Search components matching the search criteria defined by user among those described below.

According to user's preferences, the result may contain a set of single components matching the search criteria or a set of business specifications that contain at least one or more components matching the search criteria.

The search function returns a list of components or a list of Business specifications matching the selected criteria identified in Section 7.2. The search function SHOULD accept special characters and logical operators.

8.1.2.2 Consultation functions

Access to 'Consultation' functions may be restricted. (For example: users with 'submitter' role can consult their own components but can't consult others components until they're published (approved)

Mandatory functions:

- Get a component from its ID: a user can retrieve a component from its identifier whatever the component type may be. (ACC, ASCC, BCC, CCT, ABIE, ASBIE, BBIE, DataType, BusinessContext…),
- Get documents attached to a business specification,
- Get a Business Specification from one of its components (for example: get business specification from an ABIE…),
- Get an Assembly Message from one of its components,
- Get all components with the same type (for example: get all ACC, get all CCT, get all Business Contexts…),
- Get all sub-level components from a parent component (for example: get all BCC from ACC, get all Supplementary Component from Data Type…),
- Get latest, previous or all versions of a component,
- Get components or business specifications by status (e.g. get all "approved" ACCs...),
- Get all components or business specifications by owner (e.g. give me my components…).

8.1.3 Administration Interface

An Administration Interface is designed to help users with an Administrator role to perform maintenance operations on *Registry* Entries and provide suitable functions to reliably manage *Registry* content.

The Administrator role overrides all other roles permission; consequently the access to Administration Interface functionalities MUST be protected.

Administration interface SHOULD allow an Administrator to perform the following operations:

- Register/deregister new users,
- Register/deregister new groups,
- Register/deregister new organizations,
- Manage user's roles,
- Manage 'membership' association between registered users, groups and organizations,
- Add, delete or update Classification Schemes/Classifications used within Business Contexts,
- Add, delete or update CCs / BIEs stored in the *Registry* with dependencies checking and warning,
- Change components status among 'submitted', 'approved' and 'deprecated',
- Change components state among 'work in progress', 'validation', 'rejected'…
- Provide methods to check content reliability:

- Warn Administrator before deleting a referenced component,
- List broken associations,
- List deprecated dependencies,

- Provide statistics:
  - Total number of business specifications by status/owner/organization,
  - Users activity,
  - Number of queries by component/business specification/assembly message,
  - Mean time in process stages.

Additional statistics functions may be defined.

Administration interface may offer tools to schedule maintenance operations (e.g. file back-up, content restore, etc.).

8.2 Web Interface

Initially (as illustrated by RepXML, see Annex B), a web interface is designed to allow searching and viewing of *Registry* based CCs using an Internet browser (Mozilla Firefox or Internet Explorer).

The Web user interface SHALL be multilingual to facilitate ready access, at a minimum covering the UN/CEFACT working languages, English, French and Russian. The web interface uses the client application interface (Web services) to discover components and get data from the *Registry*.

Other functions will be very important for UN/CEFACT Working Groups, they will facilitate easy integration of all newly submitted Artefacts.

Submitted components will be validated by UN/CEFACT TBG Groups using multiple tools. These tools SHOULD be accessible via web interfaces and SHOULD use Client Application Interfaces. These tools will be available only to authenticated users that have been registered using the *Registry* administration interface.

The web interface will allow *Registry* Artefacts to be exchanged in multiple formats, (e.g., XML Schema NDR, UML-XMI, etc.).

9 Security

9.1 Authorization and Access Control

The *Registry* MUST support role-based Access Control (RBAC) in order to enforce the following types of access policies:

- Registered users can read, create and reference standard Business Information;
- Validation authorities can validate standard Business Information;
- Guest users (and other registries acting on behalf of guest users, in case of *Registry* federation) can only read Business Information;

- Non Business Information (e.g. Security Policies, user, identifiers roles…) MUST NOT be accessed by normal users. Such objects are managed by the *Registry* administrator;
- Authenticate the identity of any entity entitled to create, validate or reference Business Information as documented in Section 9.2;
- Assign roles to organizations, groups or users as recommended by a recognized authority or object owner. *Registry* administrators can assign any role to any user, and they can assign any access right to themselves;
- In order to prevent fraud, additional measures MUST be deployed if separation of duties is required. This is implementation specific.

9.2 Authentication

The *Registry* MUST be able to authenticate the identity of the user associated with client requests in order to perform authorization and access control as defined in Section 9.1.

This requires that the *Registry* maintain a "registry keystore" that contains the public keys of the Registered Users.

The *Registry* may also rely on a trusted third party to perform user management and authentication. The use of an external Authentication authority requires that the *Registry* implements the Registry SAML 2.0 profile as defined in [ebRS].

9.3 Integrity

9.3.1 Integrity of *Registry* Content

The content of the *Registry* MUST be protected against unauthorized modification. This is provided through Access Control and signature of *Registry* items.

It is not mandatory that the signature at the *Registry* level can be checked by the client. The integrity of the *Registry* item can be checked by the *Registry* application on behalf of the client. The data integrity is then ensured between the *Registry* and the client in a next phase as described in Section 9.3.2.

However, if the *Registry* item is conveyed between Client and *Registry* as a MIME attachment, the *Registry* item MUST be signed by the sender and verified by the receiver using the RepositoryItem Signature, separately from the Message level signature, in conformance with [ebRS].

9.3.2 Integrity of data interchange between the *Registry* and clients

The *Registry* MUST provide mechanisms to ensure data integrity during transmission between the client and the *Registry*.

This can be provided at transport layer by HTTP/S communication using SSL/TLS and/or by message layer integrity provided by [WSS-SMS].

9.4 Confidentiality

The *Registry* may provide encryption at transport layer when SSL/TLS is used. Alternatively, XML encryption can be used in order to encrypt SOAP Message bodies or attachments, if required.

9.5 Key distribution

Authentication, Integrity and Confidentiality mechanisms require an infrastructure for secure generation and distribution of public keys to the registered entities. The implementation of this infrastructure is outside the scope of this specification.

This infrastructure matches the following requirements:

- A certificate is associated with the *Registry*,
- A certificate is associated with the registered user,
- The *Registry* is enabled to acquire User Certificate and to process User Certificate verification when required,
- The Registered user is enabled to acquire *Registry* Certificate and to process *Registry* Certificate verification when required.

The details of the related procedures are implementation specific.

9.6 Single Sign On (SSO)

The *Registry* may delegate the User Identity management and User Authentication to a legacy system or to a specific trusted entity which maintains user information and credentials. This can be done by implementating support for SSO operations.

A registry enabling SSO SHOULD support the SAML profile enabling SSO. Web SSO and Liberty SSO are also eligible for provision of SSO operations.

9.7 Audit trail

The *Registry* MUST maintain an Audit trail in order to trace all access requests and operations resulting of access requests. The storage of the logged events is implementation specific. It is recommended that the storage interface is implemented in line with the relevant standards in order to facilitate co-operation between federated Registries to counteract large scale attacks.

10 Glossary

***ABIE**: Aggregate Business Information Entity.* A collection of related pieces of business information that together convey a distinct business meaning in a specific Business Context. Expressed in modelling terms, it is the representation of an Object Class, in a specific Business Context.

***ACC**: Aggregate Core Component.* A collection of related pieces of business information that together convey a distinct business meaning, independent of any specific Business Context. Expressed in modelling terms, it is the representation of an Object Class, independent of any specific Business Context.

***API**: Application Programming Interface*

***ASBIE**: Association Business Information Entity.* A Business Information Entity that represents a complex business characteristic of a specific Object Class in a specific Business Context. It has a unique Business Semantic definition. An Association Business Information Entity represents an Association Business Information Entity Property and is associated to an Aggregate Business Information Entity, which describes its structure. An Association Business Information Entity is derived from an Association Core Component.

***ASCC**: Association Core Component.* A Core Component which constitutes a complex business characteristic of a specific Aggregate Core Component that represents an Object Class. It has a unique Business Semantic definition. An Association Core Component represents an Association Core Component Property and is associated to an Aggregate Core Component, which describes its structure.

***ASN1**: Abstract Syntax Notation One*

***ATG**: UN/CEFACT Applied Technology Group*

***BBIE**: Basic Business Information Entity.* A Business Information Entity that represents a singular business characteristic of a specific Object Class in a specific Business Context. It has a unique Business Semantic definition. A Basic Business Information Entity represents a Basic Business Information Entity Property and is therefore linked to a Data Type, which describes it values. A Basic Business Information Entity is derived from a Basic Core Component. ABIE.

***BC**: Business Context.* The formal description of a specific business circumstance as identified by the values of a set of Context Categories, allowing different business circumstances to be uniquely distinguished.

***BCC**: Basic Core Component.* A Core Component which constitutes a singular business characteristic of a specific Aggregate Core Component that represents an Object Class. It has a unique Business Semantic definition. A Basic Core Component represents a Basic Core Component Property and is therefore of a Data Type, which defines its set of values. Basic Core Components function as the Properties of Aggregate Core Components.

***BIE**: Business Information Entity.* A piece of business data or a group of pieces of business data with a unique Business Semantic definition. A Business Information Entity can be a Basic Business Information Entity (BBIE), an Association Business Information Entity (ASBIE), or an Aggregate Business Information Entity (ABIE).

***BP**: Business Process*

***BPSS**: Business Process Specification Schema*

***BRS**: Business Requirements Specification*

***BS**: Business Specification*

***CBPC**: Common Business Process Catalog*

***CC**: Core Components.* A building block for the creation of a semantically correct and meaningful information exchange package. It contains only the information pieces necessary to describe a specific concept.

***CCT**: Core Component Type.* A Core Component, which consists of one and only one Content Component, that carries the actual content plus one or more Supplementary Components giving an essential extra definition to the Content Component. Core Component Types do not have Business Semantics.

***CCTS**: Core Components Technical Specification*

***DSIG**: Digital Signature.*

***DT**: Data Type.* Defines the set of valid values that can be used for a particular Basic Core Component Property or Basic Business Information Entity Property. It is defined by specifying restrictions on the Core Component Type that forms the basis of the Data Type.

***DTD**: Document Type Definition*

***ebRS**: ebXML Registry Services*

***EDI**: Electronic Data Interchange*

***HTTP/S**: Secure HyperText Transfer Protocol*

***ICG**: UN/CEFACT Information Content Management Group*

***JAXR**: Java API for XML Registries*

***NDR**: Naming and Design Rules*

***ODP**: Open Development Process*

***RIM**: (ebXML) Registry Information Model*

***RSM**: Requirements Specification Mapping*

***SAML**: Security Assertion Markup Language*

***SOAP**: Simple Object Access Protocol*

***SSL**: Secure Socket Layer*

***SSO**: Single Sign On*

***TBG**: UN/CEFACT International Trade and Business Processes Group*

***UML**: Unified Modelling Methodology*

***UMM**: UN/CEFACT Modelling Methodology*

***UN/EDIFACT**: United Nations directories for Electronic Data Interchange for Administration, Commerce and Transport*

***UN/LOCODE**: United Nations Code for Trade and Transport Locations*

***UN/TDED**: United Nations / Trade Data Elements Directory*

***UUID**: Universal Unique Identifier*

***WCM**: Web Content Management????*

***XMI**: XML Metadata Interchange*

Annexes

A. *Registry* deployment considerations (informative)

The *Registry* and UN/CEFACT compliant registries SHALL comply with the recommended version of the ebXML registry specification and all normative sections of this specification.

freebXML Registry Project

freebXML.org is an initiative that promotes open source development of ebXML-based applications. Among the different freebXML projects, freebXML Registry is a general registry adopting a generic and extensible information model that includes the ability to have arbitrary associations between entries in the registry. For more information see the web site at http://ebxmlrr.sourceforge.net.

freebXML Registry is, at this time, the most advanced and compliant implementation of ebXML Registry Service specifications and it will be used for the first implementation of the *Registry*.

Volume

This section provides an indication of target *Registry* capability in terms of transaction volumes and storage capacity. The first part gives a constraint on the number of objects that MUST be managed by the *Registry*, the second part gives some constraints on the level of transaction.

Capacity

Context management SHOULD facilitate the re-use of business contexts that have already been defined rather than declaring new business contexts. Every code added to a business context SHALL be approved by the maintenance agency.

The *Registry* vocation is to store CCs and more generic BIEs, without the obligation to provide a particular business context for every Artefact. In any case a specialized registry for business contexts could be implemented to share its stored business contexts with the federation service.

In the *Registry*, there are two different parts:

- Contexts,
- Objects, i.e. CC and BIEs.

For the Context, there are two kinds of information:

- Context values and Classification scheme,
- Business Value.

The Context values are fixed at the beginning and the evolution is very rare.

The *Registry* SHALL manage more than 1000 context values for each of the eight [CCTS] specified contexts.

The number of Business Values can be potentially huge, but normally, if the Business Values are used by exception, the number of these Business Values should not be so important. Thence the *Registry* SHALL be able to managed more than 1000 values for BP context.

For the CCs, at least, the *Registry* SHALL manage:

- 100 Core Component Types,
- 1000 Data Types,
- 1000 Aggregate CCs (and the associated Basic CCs and Association CCs).

This estimation is based on the UN/EDIFACT Directory version D.04A which is composed of:

- 145 Messages,
- 156 Segments,
- 198 Composites,
- 650 Elements.

For the BIE, at least, the *Registry* SHALL be able to manage:

- 5000 Data types,
- 10000 Aggregate BIEs (and the associated Basic BIEs and Association BIEs).

This estimation is based on different scenarios of specialisation of the CC on different countries, different Product classifications and different Industry classifications and / or different Business Roles.

Transactions

There are two kind of transaction with the *Registry*:

- Submission and modification of objects,
- Requests on the *Registry*.

The number of transactions supported by the *Registry* SHALL be at least:

- 120 submissions per hour,
- 1000 requests per hour.

For these transactions, the *Registry* SHALL reply in an acceptable time:

- 3 seconds for a request,
- A half minute for a submission.

Encoding

The encoding MUST support multilingualism (as example: UN Official Languages). In order to support multilingualism, the encoding MUST be UNICODE. It allows managing all new languages defined in ISO 639-2 normalization.

B. RepXML overview – Example (Informative)

RepXML is a project that has been launched by EDIFRANCE. Its aim is to develop a registry/repository prototype that provides a complete set of francophone business components (BIEs) that is accessible on the Internet.

More specifically, the objectives of the project are as follows:

- Define and implement the procedure that enables business groups to submit, retrieve and re-use e-business components (francophone BIEs),
- Implement a registry/repository that registers and stores these components,
- Develop a "Connector" that allows business groups to create off-line new BIEs,
- Develop a public Internet site,
- Develop the interface between the Connector and the Reg/Rep server,
- Conduct a field trial with EDIFRANCE members.

This prototype will enable Business Groups to submit Business Specifications that will be controlled and validated by a validation authority. The validated BIE is then published in the registry/repository.

The user can access the service either by the RepXML web site for consultation, or via a connector.

A connector is a stand alone client application that is hosted in the local Information System of the user. It provides the tools that enable the user to prepare the Model off-line.

This prototype will be linked with the UN/CEFACT *Registry* that will contain the CCs.

Standardisation compliance

Several specifications are being considered during the RepXML project. These specifications are listed below:

CCTS 2.01: CCTS constitutes Part 8 of the ebXML Framework. This specification defines building blocks, either context neutral (CCs) or context specific (BIEs), that are stored in an ebXML registry.

UML 1.5: UML (Unified Modelling Language) is a specification defining a graphical language for visualizing, specifying, constructing, and documenting the artefacts of distributed object systems. UML being the essential component of the UMM, all library components (CCs and BIEs) shall be expressed in UML.

ebXML Registry 2.5: The ebXML Registry Services provides a set of services that enable sharing of information between interested parties for the purpose of enabling business process integration between such parties based on the ebXML specifications.

The shared information is maintained as objects in a repository and managed by the ebXML Registry Services.

The [ebRIM] provides information on the type of metadata that are stored in an ebXML registry as well as the relationships among metadata classes. The [ebRIM] defines in a generic manner what types of objects are stored and defines how stored objects are organized, but it does not specifically address the storage of CCs and BIEs.

A syntax neutral model and a format based on which must be defined allowing CCs and BIEs to be bound to the registry.

General principles

RepXML is a registry/repository where business components are registered and stored. These business components can be composed of BIEs, XML Schemas and/or business documents. In the remaining part of this Annex, these business components are called *Business Specifications*.

The main purpose of RepXML is to allow users to prepare and submit to the server new proposed *Business Specifications* for validation, and to retrieve validated *Business Specifications* from the server.

RepXML can be accessed via a specific software application (called the Connector) or via a Web interface. Using the connector a user can prepare off-line a new proposed *model* using some sort of work sheet, and then submit this work sheet to the server.

For clarification, it is convenient to distinguish between several categories of users interacting with RepXML:

- Author: a person that creates a new BS (eg for the needs of a particular business sector),
- Submitter: a person or an organisation that submits a newly created BS,
- Validator: a person or a group of persons from a validation authority that validates the new proposed business components (eg UN/CEFACT),
- Guest user: a person from a company or an organisation that retrieves validated business components for their specific needs,
- Administrator: a person from the organisation in charge of managing and maintaining the server.

Architecture / modules

The overall functional architecture is shown in Figure 25. The two main modules are the Connector and the RepXML application.

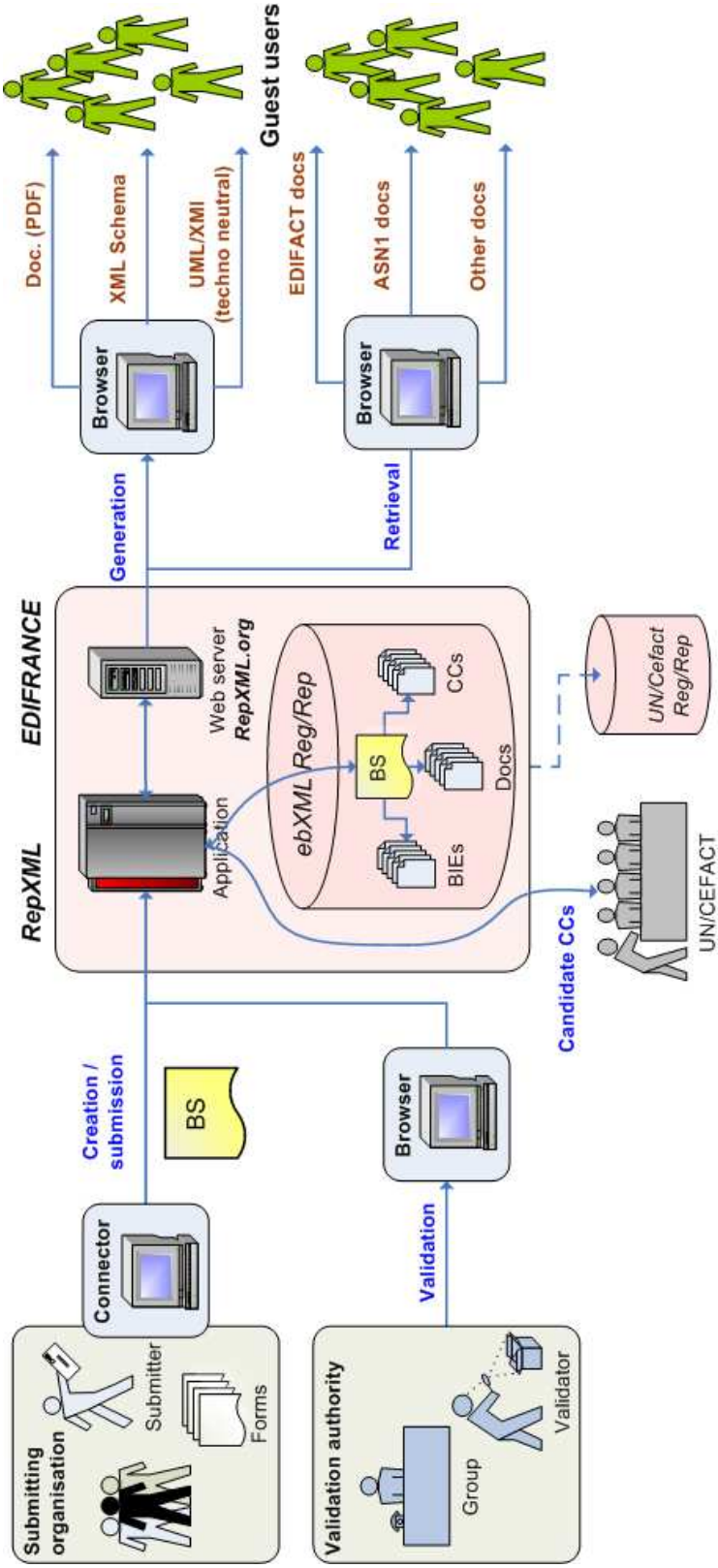

Figure 25 - RepXML functional architecture

Connector

The connector is a piece of software that allows submitting organisations to create BSs and to submit these to a validation authority via the RepXML application. Before creating a BS, one has the possibility to perform a semantic search on the valid BIEs, either locally in the data base attached to the Connector or in the registry of the remote RepXML application.

When submitting a BS to the RepXML registry, a specific submitting procedure is started which performs some consistency control before the actual sending out.

BS forms can be downloaded from the RepXML registry to the connector on demand or in a systematic manner so as to update the BS forms that are locally present in the connector.

The connector may also be used by guest users to retrieve from the RepXML registry pieces of information in different formats: either existing documents (eg PDF, EDIFACT, ASN1…), or automatically generated documents (UML models of BIEs in PDF or XMI formats, or XML Schemas of BIEs).

RepXML application

The RepXML application is based on existing components like the ebXML registry/repository (freebXML) and a key registry (keystore). These components are interfaced with the RepXML application without being modified. The ebXML registry/repository provides a set of methods that are accessible to the RepXML application for storing, searching and organising objects.

A Web Server (RepXML.org) is interfaced with the RepXML application. It allows guests users (development teams, IS designers…) to access and visit the Web site, and to retrieve RepXML BSs in different formats.

freebXML

freebXML.org is an initiative that promotes open source development of ebXML-based applications. Among the different freebXML projects, ebXML registry/repository is a general registry adopting a generic and extensible information model that includes the ability to have arbitrary associations between entries in the registry. The repository is a way of storing information about entries in the registry. Any type of data can be stored in the repository.

The ebXML registry/repository is used to reference and store common pieces of business data thus allowing users to compose B2B messages.

RepXML.org Internet Web Site

The RepXML.org is a public Internet site. The site offers:

- A multi criteria research,
- To view and download all public business components, and the associated documents,
- To export the components in different format, like XML, PDF, XMI, Xform…

- To submit, but not to edit, a new business component,
- Member based access,
- A discussion forum with a news space.

Submission / validation procedures

Submission procedure

This procedure describes the sequence of actions/exchanges between the submitting organisation and the RepXML application. There are three steps in this procedure:

- Synchronization between Connector and RepXML,
- The author creates a new model,
- Model is submitted to RepXML.

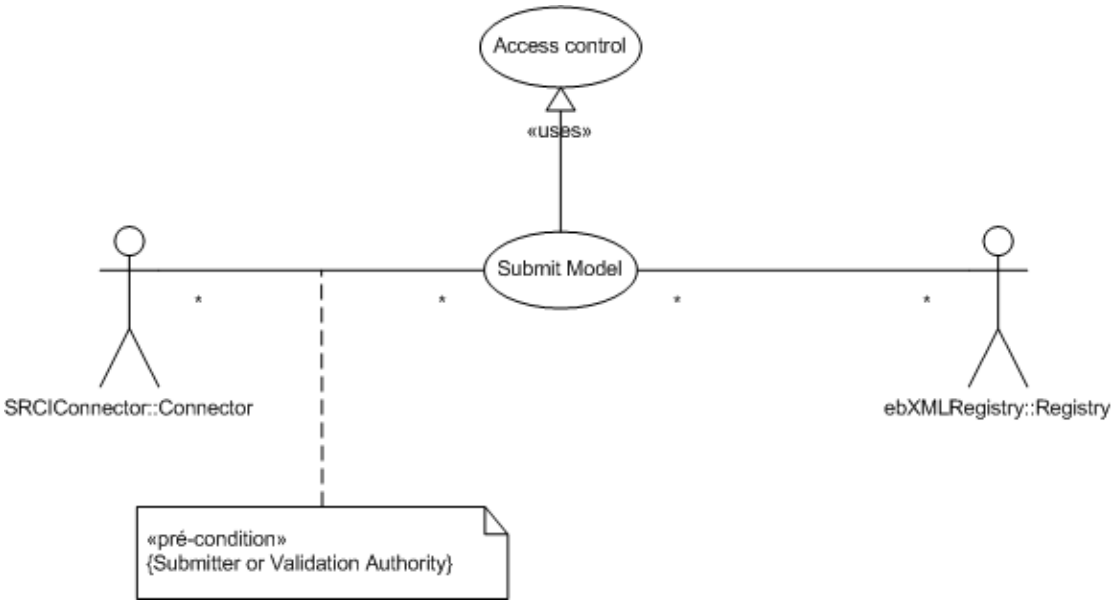

Figure 26 - Submit Model use case

Validation procedure

This procedure describes the sequence of actions/exchanges between the validation authority and the RepXML application. There are four steps in this procedure:

- Validator is informed that there has been a new model submitted,
- Validator evaluates new model,
- Validator validates or rejects (with reasons). In the case where the Validation Authority thinks that the submitted model requires the creation of new CCs, then there is an additional step in this procedure,
- Candidate CCs proposed by Validator to the UN/CEFACT.

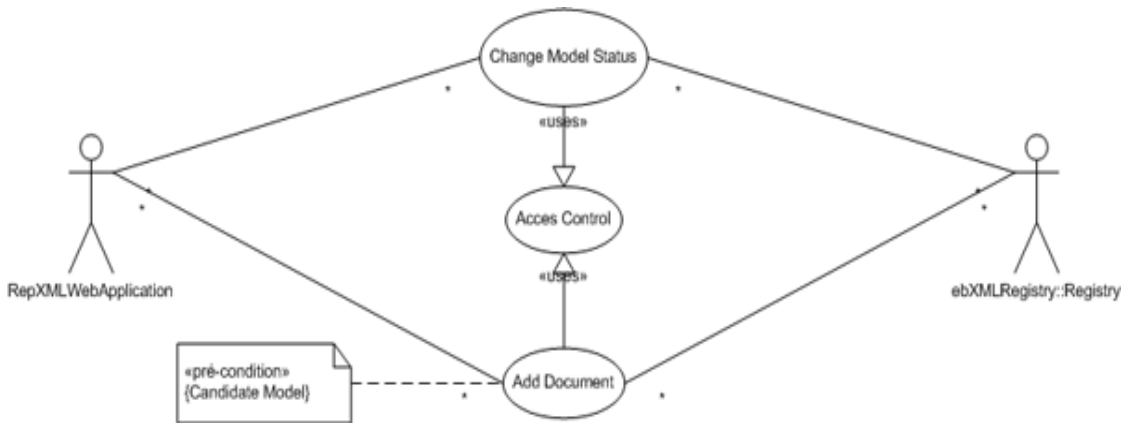

Figure 27 - Validate Business Specification use case

Storing CCs/BIEs in RepXML

Regarding the storage of CCs and BIEs in the RepXML registry, we have used the same metadata approach as Republica in the LomakeFi registry initiative, that is CCs and BIEs are stored as registry entries. However we also have the possibility to store items such as documentation (eg in PDF) in the Repository.

Storing CCs and BIEs follows a specific BS based on the ebXML Registry Information Model (v2.5). The details of this model very similar to the one used by Republica are as follows:

CCTS Classes Mapping to the ebXML RIM Objects		
CCTS Object	**RIM Object**	**RIM Object Type**
BS	ExtrinsicObject	Model
ACC	ExtrinsicObject	AggregateCoreComponent
BCC	ExtrinsicObject	BasicCoreComponent
CCT	ExtrinsicObject	CoreComponentType
ASCC	Association	AssociationCoreComponent
ABIE	ExtrinsicObject	AggregateBusinessInformationEntity
BBIE	ExtrinsicObject	BasicBusinessInformationEntity
BC	ExtrinsicObject	BusinessContext
ASBIE	Association	AssociationBusinessInformationEntity
DT	ExtrinsicObject	DataType
The 8 Context Categories of BC	ClassificationSchemas	
Contexts	ClassificationNode	
CCTS Classes structure Mapping to the ebXML RIM associations		
CCTS Object source	**CCTS Object target**	**RIM association type**
ACCs	BCCs	Contains
ABIEs	BBIEs	Contains
BBIEs	DTs	HasType

DTs	CCTs	Refine
CCTs	Content Components	Contains
CCTs	Supplementary Components	Contains
BIEs	CCs	Refine
BC	BS	ContextFor

CCTS attributes Mapping to the ebXML RIM attributes		
CCTS attribute	**RIM attribute**	
Dictionary Entry Name	RegistryObject Name	
Definition	RegistryObject Description	
all the other fields	Slots	

A Java API developed for this model runs on top of JAXR (Java API for XML Registries).

Currently a BIE only points to the identifier and name of its corresponding CC. This is a pragmatic approach that allows us to store BIEs without having to wait for CCs to be standardised. CCs are stored in the registry, and the storage of dependant BIEs adjusted as necessary.

Figure 28 provide an example of how the Person Model is stored in the registry.

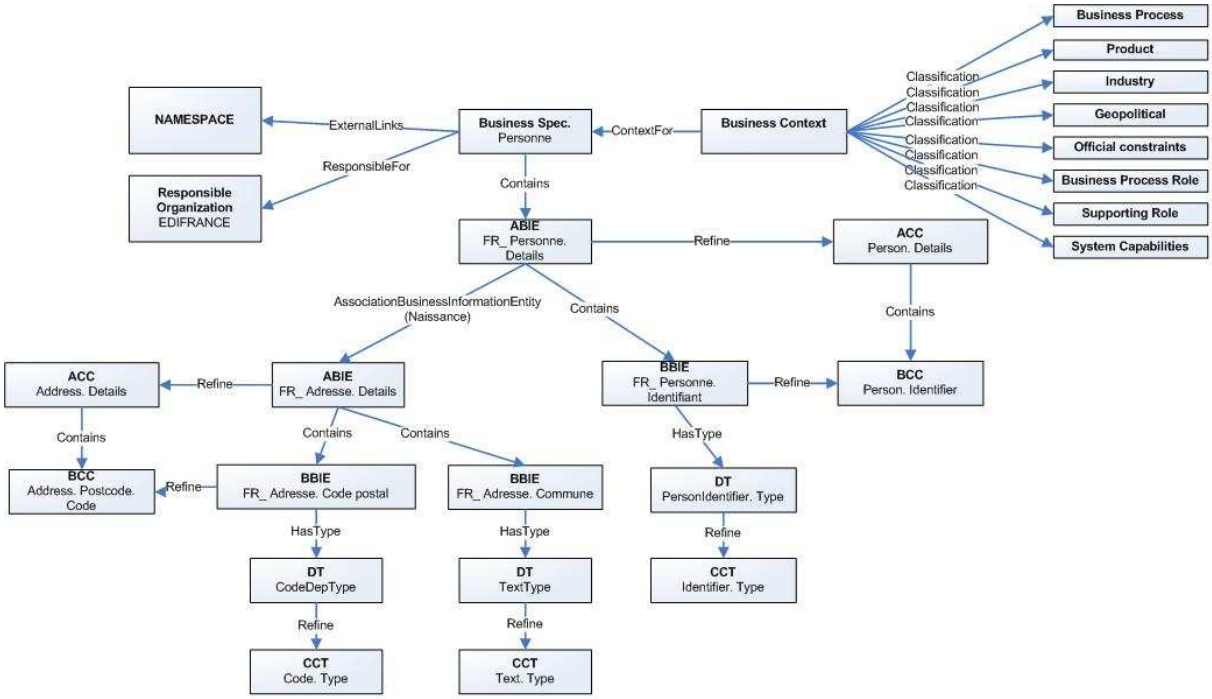

Figure 28 - Stored Person Business Specification in the RepXML registry

Contact Information

Team Leader:

Name	Francis Berthomieu
Company	France Telecom
Street	42 Rue des Coutures
City, state, ZIP/other	Caen, 14000
Nation	France
Phone	+33 (0)2 31 75 91 30
E-mail	francis.berthomieu@orange-ft.com

Editors:

Name	Ivan Bedini
Company	France Telecom
Street	42 Rue des Coutures
City, state, ZIP/other	Caen, 14000
Nation	France
Phone	+33 (0)2 31 75 95 41
E-mail	ivan.bedini@orange-ft.com

Name	Fabrice Bourge
Company	France Telecom
Street	42 Rue des Coutures
City, state, ZIP/other	Caen, 14000
Nation	France
Phone	+33 (0)2 31 75 91 45
E-mail	fabrice.bourge@orange-ft.com

Copyright Statement

Copyright © UN/CEFACT 2006. All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to UN/CEFACT except as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by UN/CEFACT or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and UN/CEFACT DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.