Creating A Single Global Electronic Market

Registry/Repository

Program Interface Access Specification

Draft, Version 0.21, 18 September 2000

Working Document.

Abstract

The need is to define both the classification system and the associated interface semantics for Registry/Repository as program level interfacing via XML structures and methods to the business semantic information definitions.

Status

This draft represents the blending of current practical work in a variety of areas with XML, including the latest W3C Schema and Datatyping drafts, ISO11179, OASIS Registry and IETF WebDav DASL work.

Contributors

Document Editor: TBD

Editor:

David RR Webber

Review Team:

Farrukh Najmi, Len Gallagher, Michael Kass, Scott Nieman, Bruce Peat.

1. Table of Contents

Creating A Single Global Electronic Market

2. Introduction

The objective of this document is to provide the necessary details for an understanding and specification details of the classification and interfacing to business process semantic information stored in an ebXML compliant Registry/Repository.

The top level is the *classifications*. This mechanism allows you to group together industry vertical sets of transactions so you can quickly and easily find the particular business functional components that you require based on business use and context. Classification structures then allow access to the specific low-level semantics of the business definitions and rules.

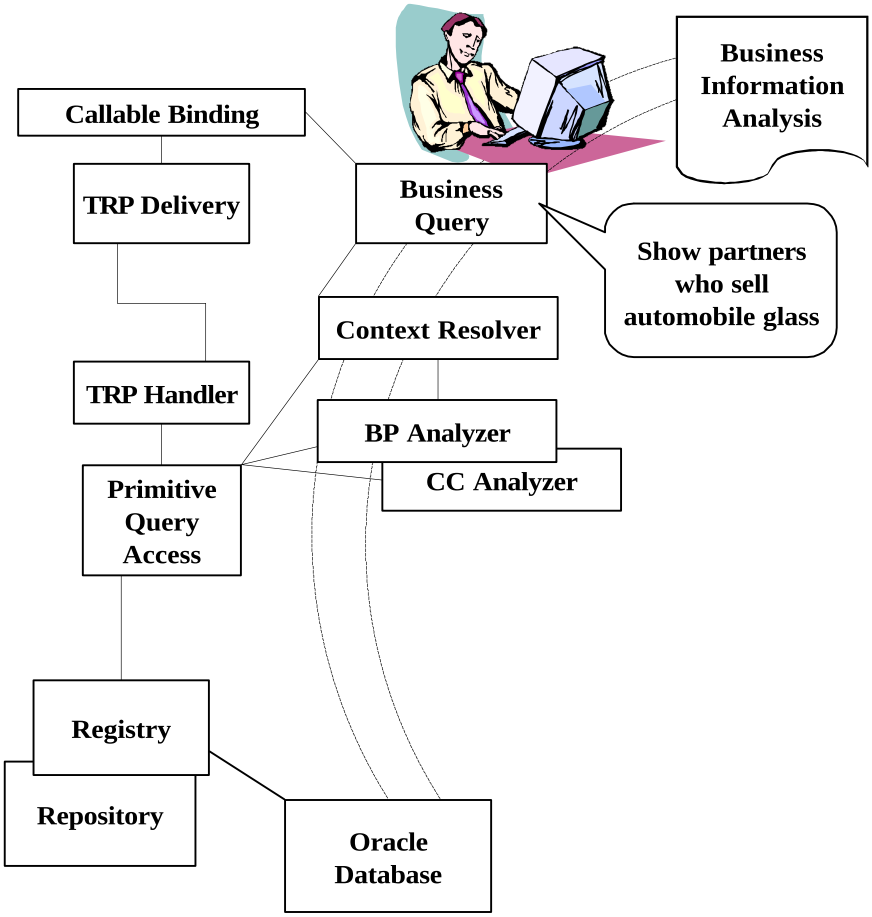
The interface specifications then show how those low-level semantics are stored, accessed and retrieved for use.

2.1 Business Use Models and Requirements

The following diagrams show how the various business use models and requirements are met with the appropriate implementation architecture. These also show the interaction models and exchanges of information that are required.

The first diagram shows a generalized application information access model and associated requirements. This document is not intended to specify the requirements and interchanges that this illustrates. It is provided here as a means of distinguishing the scope of this document from the overall scope determined for all Registry/Repository implementations. This first figure therefore shows a datawarehouse style information deployment where the Registry/Repository is essentially acting as the data dictionary and table directory that exists today in a RDBMS or OODBMS deployment. This information store is then accessed via a TRP transport compliant delivery mechanism.

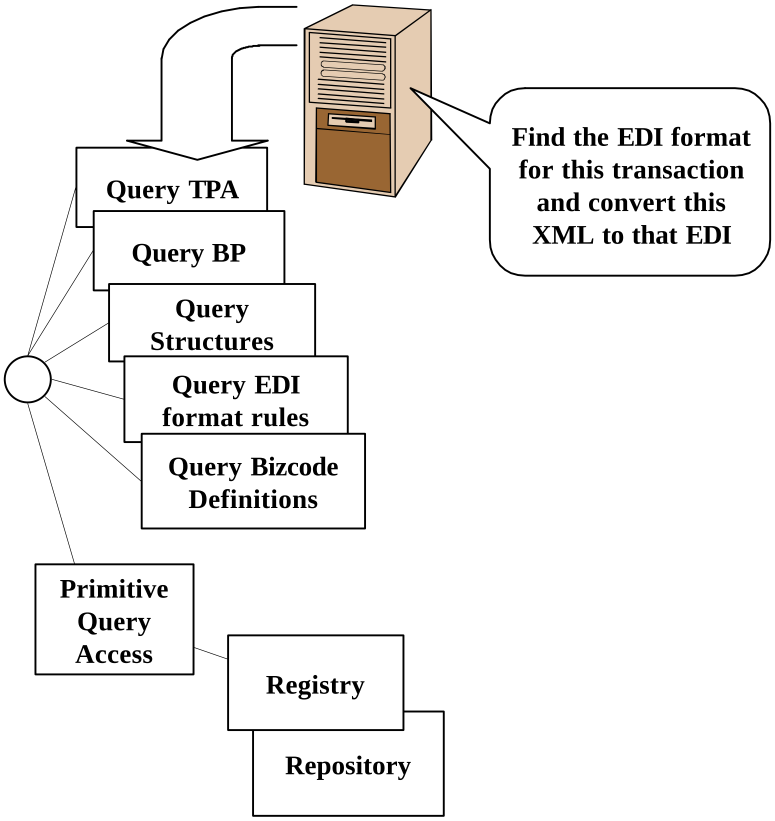
Figure 1.   Application information (datawarehouse) interaction model



The figure also shows how the high level business application query "show partners who sell automobile glass", must cascade through a series of low-level direct primitive queries to resolve the context and business semantics of the actual database in order to issue the appropriate query. The information from this query then flows from the Oracle database to the end-user application via TRP compliant delivery layer (callable bindings can hide the physical implementation layer). From the transactional stance this whole interaction uses TRP as a means to deliver a transaction payload (in this case the query) and then receive a TRP response some time later, with the application results as a response payload. To all intents and purposes this functionality mirrors that familiarly found in a database transactional monitor system such as BEA Tuxedo™, coupled with the ability to define an object hierarchical model of the information store structures across potentially multiple such information stores.

The next figure shows the opposite. Instead of a user directed query, the system is handling a set of discrete requests for low-level semantic information to resolve a transformation of business semantic content from one structural format to another (in this case, convert XML to and EDI format). The transformation is dependent on the specific trading partner and business process, and so the machine interface must retrieve this reference semantics as XML structures. Such structures must have an amount of predictable structure to them to allow a deterministic programmatic access to the rules and definitions. Part of the role of ebXML is to define those base primitive structures that essentially bootstrap any one particularly industry vertical being able to consistently store their own definitions and usage.

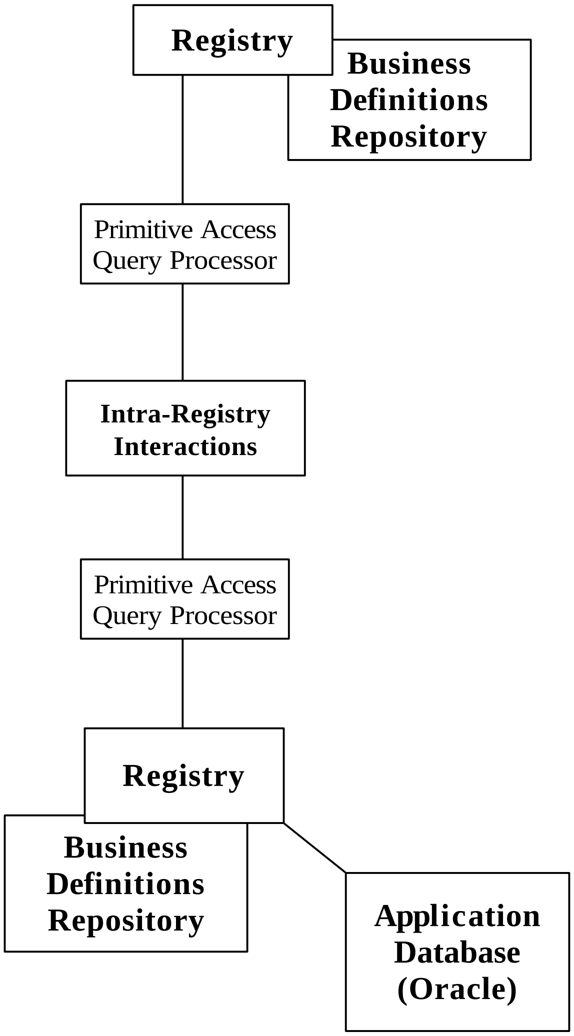
Figure 2. Machine directed semantic and primitive content retrieval


Find the EDI format for this transaction and convert this XML to that EDI
Query TPA
Query BP
Query Structures
Query EDI format rules
Query Bizcode Definitions
Primitive Query Access
Registry
Repository

The requirements for this level of interaction are quite different from the application level in Figure 1. A set of discrete interfaces to each layer of the ebXML information matrix, namely TPA, BP/CC and legacy EDI context (such as are defined at www.igML.org) are required.

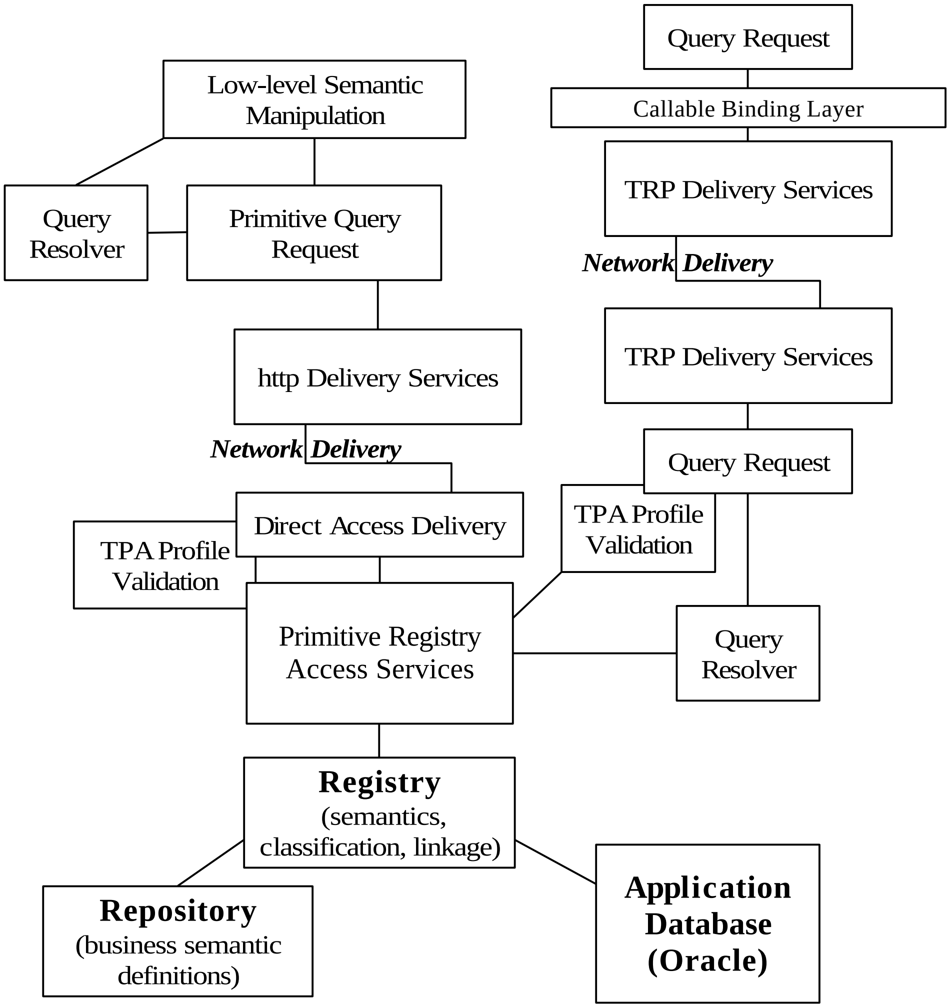
In this context interactions maybe needed between registries in a networked environment. For instance, a registry may resolve a query for EDI igML definitions by remotely querying those from a Registry that specializes in only that information. The next figure shows the major interaction component requirements for that interaction model.

Figure 3. Registry-to-Registry query interfacing.


Registry
Business Definitions Repository
Primitive Access Query Processor
Intra-Registry Interactions
Primitive Access Query Processor
Registry
Business Definitions Repository
Application Database (Oracle)

This figure shows an optional application database also; to illustrate that application information may also be resolved this way also. The next figure then combines all the interaction models to show how both TRP and Registry primitive access are combined together in order to fully meet all the requirements.

Figure 4. Registry interaction mechanisms and architecture model.



This figure shows how all interaction models relate. However the focus of this specification document is on only the Registry Primitive Access Services. This focus is dictated both by the requirements identified for the Tokyo PoC applications, and also as an assessment of the broader use need. Clearly the higher-level application information usage requirements and model cannot be implemented until the base level primitive mechanisms can store and retrieve business process, core component and reference table information.

2.2 Design Goals

The ebXML principles require that the Registry primitive access services XML syntax used must be:

1) Simple to understand, to learn, read and use.

2) Provide a concise feature function set thereby ensuring consistent implementations, interoperability, and low cost of adoption. Each feature must earn its place based on widespread business need and applicability.

3) Separate the query, change and representation syntax, and use existing work such as IETF WebDav DASL wherever possible.

4) Support the storage and retrieval of ebXML Business Process and Core Component definition methods.

5) Provide a human interface for information discovery via a direct browser form based interactions and allowing rendering with multilingual support.

7) Provide a simple metaphor to migrate and express existing data dictionaries and related content such as COBOL copybooks, SQL table definitions, CICS structures, program data structures, business data dictionaries and similar information content quickly and easily into.

8) Be based on the W3C XML markup syntax, with minimal use of extended features, and be consistent with and interoperable with the ebXML technical specifications.

9) Above all, provide both large industry partners and small businesses with mission critical high volume, high performance, and open public standard based interchanges. Coupled with the long term means to conduct and maintain cost effective electronic information exchanges that can be simply deployed and exploited by as large a cross-section of the workforce as possible.

2.3 Terminology and Concepts

The following extracts are provided to aid understanding of this document.

2.3.1 Classification

A classification is a partition of a given collection of items into mutually exclusive and collectively exhaustive sub-collections. A classification depends upon a pre-existing specification of a hierarchy of values, names, and codes called a classification scheme. Registry items in a Registry may be classified by as many classification schemes as deemed appropriate by the Submitting Organization. A classification scheme can have an associated XML structure that defines the information within the classification. An example would be currency table that has currency code, currency symbol, name, country code, conversion rate and date associated with it. Classifications may be referential; so one classification may depend on another classification.

A distinction can therefore be made between classifications that describe physical business content as above, and classifications that describe collections of like information within the registry itself, such as XML structure layouts associated with business processes.

3.3.1 Coded Classification Scheme

A coded classification scheme is a hierarchy of values that can be referenced by a classification. A coded classification scheme can vary from a simple set of values to a complex multi-level hierarchy. An example of a simple single-level coded classification is the set {Freshman, Sophomore, Junior, Senior} used to partition a collection of students. An example of a more complicated classification scheme is one based on the hierarchy of all living things with named levels for Kingdom, Phylum, Class, Order, Family, Genus and Species.

4.3.1 Package

A Package is a conceptual notion used to identify a set of registered objects. It is defined to be a registered object that is a set of pointers to other registered objects. Using this definition, a package can represent a hierarchy of registered objects, where non-terminal nodes of the hierarchy are other packages and terminal nodes are package or non-package objects. A package is a terminal node in a package hierarchy if and only if the package is empty. A registered object may be pointed to by several different packages. A package relationship between a registered package and some other registered object pointed to by a package element is represented by the *contains* role in an association instance.

Since the representation of a registered object is defined to be a file, the file representing a package object is an XML document.

5.3.1 Query

A query is a message from a public user of a registry database to a registry, asking that certain information be returned. A request is sent in the form of an XML document that validates to one of the XML query DTD's defined elsewhere in this specification. The response to a query will validate to the associated XML response wrapper DTD.

6.3.1 Change Request

A request is a message sent from a Submitting Organization to a Registration Authority asking that certain additions or modifications be made to the Registry. A request is generally sent in the form of an XML document that validates to one of the request DTD's defined elsewhere in this specification. A request instance will consist of a request code to identify the type of request as well as the XML content of a specific request.

Further details on the terminology definitions can be found from the OASIS Information Model document, and the ebXML Part 1 Repository specifications document.

2.4 Relationship of Information Model

The objective is to provide layers of XML classification syntax for the ebXML functionality of TPA, BP and CC, a legacy EDI data dictionary, TRP and any directly associated content such as UDDI that naturally overlay onto the classification system required by an ebXML compatible Registry system. Once such approach here is the ebXML GUIDE classification system (http://www.xmlguide.org).

Similarly an ebXML compatible registry change or query request can then be mapped into an existing classification XML structure. Such change or query requests can then be easily structured relative to the XML structure using WebDav style DASL querying mechanisms.

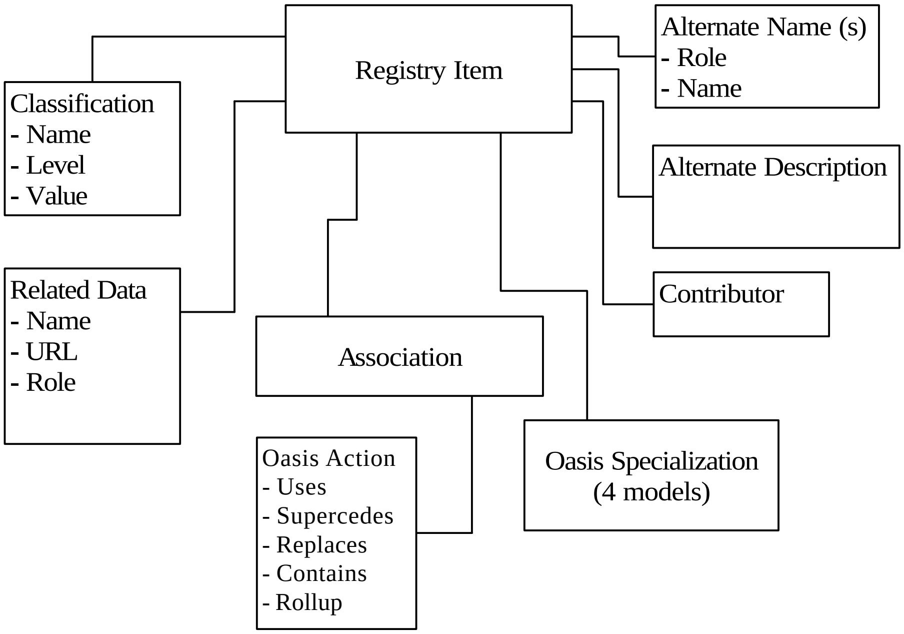
Further work is underway to similarly provide a bridge to an ISO11179 compatible repository at the level of the element definition layer.

The following figure illustrates the Registry classification model expressed as an OASIS information model. For ebXML the classification syntax noted above: TPA, TRP, BP/CC/EDI (GUIDE), and UDDI each constrain the content information model to discrete sets.

The difference is therefore that the OASIS design is a generalized information model, while the ebXML is designed for business transactional information use and is therefore optimized to provide those interactions.

Also ebXML Registry/Repository has extensions and transformation support that OASIS registry does not provide.

Figure 5. OASIS Registry Information Model



For more extended information on the OASIS registry specifications please see http://www.xml.org and associated content. Also see Registry/Repository Classification Specifications document.

2.5 Attribute Types

Attribute values in the information model will be one of the following types:

- Entity References
- Base Types

Some attribute values will be references to entity instances and some will be primitive types that can be represented as character strings, numbers, dates, or dates and times. Identified entity references include one of the following types:

REGISTRY_ITEM
ORGANIZATION
CONTACT
SUBMISSION

To this list we add the Enumeration Entities defined below.

The following definitions identify the base types that will be used in this specification.

CodeText (valid XML tag name or reference URI) -- a character string consisting entirely of visible characters from an implied character set. The presence of non-visible characters, even blank spaces, is an error. In XML environments, CodeText may not contain XML characters with special meaning. These include the ampersand (&), etc.

ShortDescription -- a character string consisting of visible characters from an implied character set, together with optional use of blank spaces. Any other non-visible characters are ignored during processing, and other non-visible characters are stripped out before acceptance as a value of an attribute having this datatype.

Date -- a value that represents a calendar date, constrained by the natural rules for dates using the Gregorian calendar. A Registry will be able to respond to queries involving minimal date arithmetic, e.g. finding all instances of an entity having dates for a given attribute that fall within a given range, or finding all instances having dates in the past 30 days, or finding all registry items whose registration is scheduled to expire in the next 3 months, etc. More advanced date arithmetic or date manipulation is at the discretion of the Registry.

Date Literal -- a character string value that identifies a specific date. A date literal string is of the form YYYY-MM-DD where YYYY is an integer literal for the year, MM is an integer literal for the month of the year, and DD is an integer literal for the day of the month. Whenever a date value is presented to a user, or requested from a user, the date value is presented or transmitted as the equivalent date literal.

Datetime -- a value that represents a calendar date and a time within that date, with time precision to the minute, or finer. Unless otherwise indicated time is Universal Coordinated Time based on a 24-hour clock. A Registry has the capability to convert a Datetime type to a Date type, with the expected loss of precision. Any other datetime arithmetic or datetime manipulation is at the discretion of the Registry.

Datetime Literal -- a character string value that identifies a specific datetime. A datetime literal string is of the form YYYY-MM-DD HH:MM:SS where YYYY is an integer literal for the year, MM is an integer literal for the month of the year, DD is an integer literal for the day of the month, HH is an integer literal for the hour (assuming 24-hour clock), MM is an integer literal for the minute within the hour, and SS is an integer literal for the second within the minute. Whenever a datetime value is presented to a user, or requested from a user, the datetime value is presented or transmitted as the equivalent datetime literal.

SmallInt -- A non-negative integer with value less than 2**16.

URNref -- a character string that conforms to the format of a Uniform Resource Name (URN) as specified by IETF RFC 1241. The length of a URNref string is less than or equal to 150 characters. (See http://www.ietf.cnri.reston.va.us/rfc/rfc2141.txt?number=2141)

URLref -- a character string that conforms to the format of a Uniform Resource Locator (URL) as specified by W3C. The length of a URLref string is less than or equal to 150 characters. (See http://www.w3.org/Addressing/URL/5_BNF.html)

FTPref -- a character string that conforms to the format of a File Transfer Protocol (FTP) Uniform Resource Locator (URL) as specified by W3C. The default user name is "anonymous". The length of an FTPref string is less than or equal to 150 characters. (See http://www.w3.org/Addressing/URL/5_BNF.html)

FILEref -- a character string that is a URLref or an FTPref.

MIMEtype – a character string that identifies a MIME type, as listed in the official list of all MIME media-types assigned by the IANA (Internet Assigned Number Authority). The length of a MIMEtype string is less than or equal to 150 characters. (See ftp://ftp.isi.edu/in-notes/iana/assignments/media-types/media-types)

LanguageId -- a character string that identifies a human language and a country where that language has evolved. In general, it is of the form "xx-CC", where xx is a two character code (lowercase) for a human language and CC is a two character country code. Legal strings are specified by Language Identifier, definitions [33] through [38] in W3C XML 1.0. (http://www.w3.org/TR/REC-xml#sec-lang-tag).

CharEncoding -- a character string that identifies the encoding of a character set. It is specified by the encoding name (EncName) of an Encoding Declaration, definition [81] in W3C XML 1.0. (http://www.w3.org/TR/REC-xml#charencoding).

2.6 Enumeration Entities

Many of the attributes declared to be of type CodeText will have an additional constraint that the CodeText value match a specific value from a pre-defined list of values. The Registry information model represents such lists as entities with a fixed number of entity instances. We define such entities to be enumeration entities.

3.6.1 DefinitionSource

SourceCode	SourceName	Description
EbXML		Author of the ebXML Registry/Repository specification.
IEEE_LOM	IEEE Learning Technology - Learning Object Model	Author of the IEEE LOM Registry specification.
IMS		Author of the IMS Registry specification.
OASIS	Organization for the Advancement of Structured Information Standards	Author of the OASIS Registry/Repository specification.

4.6.1 PrimaryClassification

Source	Code	Name	Description
ebXML	defn	Definition	An XML definition document.
ebXML	inst	Instance	An XML instance document.
ebXML	pkg	Package	A package of registered items.
ebXML	other	Other (mimetype)	Binary content, must be related to a registered item.

5.6.1 SecondaryClassification

Items within definition and instance may be of related XML types such as XSL, xhtml and so forth. The default is XML, but MIMETYPE as an attribute may be used to qualify the exact content. Only content labelled by an applicable MIMETYPE will be accepted. An ebXML registry may choose to limit or validate MIMETYPE content, as it requires.

2.5.1 Submission Semantic Rules

1. The RegistryItem entity represents the set of all registered objects in the Registry. Each instance identifies a single registered object. A registry item instance holds only

some of the metadata for a registered object; other metadata is held by other entities in the Registry.

2. Each registry item instance is assigned a unique identifier by the Registration Authority (RA). This implicit value is said to be of type REGISTRY_ITEM. It is used to represent relationships of this instance with other information in the Registry.

3. The AssignedURN name is created and assigned by the RA. It is created to be unique within a conforming Registry/Repository implementation. When a Submitting Organization (SO) makes a submission to the Registry, it provides a local reference name of type CodeText. If possible, the RA uses that name to construct the AssignedURN.

4. The CommonName is provided by the SO.

5. The Version is provided by the SO. It can have an arbitrary format and is used only to help distinguish one registry item from another having the same common name. The AssignedURN will be different for different versions.

6. The ObjectLocation is a URL that identifies the location of the registered object. If the RA is also a repository for the item, then the RA will download the item, store it in the Repository, and create an http-based locator as a value for ObjectLocation. If the Registry is not also a Repository, then the ObjectLocation is provided by the SO and the RA has no further responsibility. The SO may also qualify the content with an AccessChannel. The ObjectLocation URL may need to be supplemented with channel and password information before the file containing the object can be retrieved. An ebXML Registry may then distinguish access to information within itself by utilizing AccessChannel rights, and assigning users to particular access channels.

7. The DefnSource takes its value from the DefinitionSource enumeration entity that identifies a collection of accredited Registry/Repository development organizations. If the Registry claims conformance to the ebXML Registry/Repository, then the DefnSource is ebXML.

8. The PrimaryClass is provided by the SO and takes its value from the PrimaryClassification enumeration entity. If the DefnSource is ebXML, then PrimaryClass identifies an element of the set {Definition, Instance, Package, Other}.

   a) The SecondaryClassification is provided by the SO and takes its value from the enumeration entity and must be a valid MIMETYPE.

The RelatedType is provided by the SO and takes its value from the RelatedDataType enumeration entity.

9. The RegStatus is provided by the RA with its value taken from the RegistrationStatus enumeration entity. For ebXML registrations, that entity includes the values {Baseline, Submitted, Registered, Superseded, Replaced, Withdrawn, Expired}. The StatusChg attribute is the datetime that the RA last approved a change for RegStatus.

10. The Stability attribute is provided by the SO with its value taken from the Stability enumeration entity. For ebXML registrations, that entity includes the values {Static, Dynamic, Compatible}.

11. The ExpiryDate is assigned by the RA upon suggestion from the SO. Some RA's may follow very definite procedures for the length of time an object can remain registered before an affirmation or withdrawal action is required. If the Expiration date passes without an SO action, then the RA initiates an expiration action.

12. The Description is provided by the SO.

13. The SubmittingOrg identifies the organization submitting the registered object. It points to a unique instance of the ORGANIZATION entity. On presentation of this information, the RA substitutes the CommonName of the organization. The SO must be known to the RA before it can make submissions to the Registry/Repository, and they each know of a unique URN for the other. The process for becoming known is not part of this specification.

14. The ResponsibleOrg identifies the organization responsible for the formal specification of the registered object. It points to a unique instance of the ORGANIZATION entity. The RO may be a formal accredited standards development organization or it may be the SO. On presentation of this information, the RA substitutes the CommonName of the organization.

15. The PublicComment may be suggested by the SO, but it is supplied by the RA. In most cases the comment will explain some administrative process that cannot be clearly determined from the standardized information. For example, this comment may explain how long the metadata for a replaced or withdrawn object remains available, or how long an expired object remains available before it is deleted.

6.6.1 AssociationType

Source	Code	Name	Description
ebXML	contains	Contains	Given item is a package that contains the associated item.
ebXML	related	Related	Given item is related to associated item and provides supplemental information for the associated item.
ebXML	supersedes	Supersedes	Given item supersedes associated item.
ebXML	uses	Uses	Given item uses associated item.

7.6.1 ContactAvailability

Source	Code	Name	Description
ebXML	Priv	Private	Contact available only to SO and RA.
ebXML	Prot	Protected	Contact available only to RA's.
ebXML	Pub	Public	Contact available to all users of registry.

2.7.1 Structure

Attribute Name	Attribute Type	Presence
AssignedURN	URNref	Mandatory
CommonName	ShortName	Mandatory
Version	CodeText	
ObjectLocation	FILEref	
DefnSource	CodeText	Mandatory
PrimaryClass	CodeText	Mandatory
SubClass	CodeText	
RelatedType	CodeText	
MimeType	MIMEtype	Mandatory
RegStatus	CodeText	Mandatory
StatusChg	Datetime	Mandatory
Stability	CodeText	Mandatory
PayStatus	CodeText	Mandatory
ExpiryDate	Date	Mandatory
Description	DescriptionText	Mandatory
SubmittingOrg	ORGANIZATION	Mandatory
ResponsibleOrg	ORGANIZATION	Mandatory
PublicComments	CommentText	

2.7.2 Semantic Rules

1. The RelatedData entity represents the set of non-registered objects that are related to registered objects. Each instance is a pairwise relationship between a single registered item and a single related data item. A registered item may map to many related data items.

2. Each instance of RelatedData depends upon a RegistryItem instance. This dependency is represented by an implicit value, RAitemId, of type REGISTRY_ITEM.

3. The DataName attribute is provided by the SO. It is intended that this be the link name for the DataLocation if related data items are presented visually to a user.

4. The DataLocation is provided by the SO. This link is not under the control of the RA and it may point anywhere. The RA is under no obligation to ensure that the link is a valid one.

5. The RelatedType is provided by the SO and takes its value from the RelatedDataType enumeration entity. It may include values not defined by OASIS.

6. The MimeType is provided by the SO. It identifies the MIME type of the related data item. The RA is under no obligation to ensure that the declared MimeType type is consistent with the actual file type of the file referenced by DataLocation.

7. The Comment is provided by the SO. It may further explain the relationship between the related data instance and the registry item it is linked to.

2.7 Default Classification Structures

The ebXML Registry is pre-loaded with a set of default classification structures. These fall under two categories. The first category covers the ebXML components such as ebXML TRP, TPA, BP/CC and the Query/Response DASL mechanisms themselves. The second category covers supporting and reference domains as elements that are basic primitives that underpin the TRP, TPA and BP/CC definitions themselves. From these basic building blocks the ebXML Registry can then accept further business domain definitions and content.

3. Registry Interfacing Models

3.1 Relation to IETF WebDav DASL work

Generally speaking the ebXML approach is to follow the DASL approach and provide a focused subset of a business functional feature set based on those technology neutral technical specifications (see http://www.webdav.org for more details). The WebDav DASL approach provides an ideal widely supported lightweight XML based interaction model. While the use of DASL is not mandated, the use of DASL as a reference implementation provides ebXML with the means to rapidly define a viable specification.

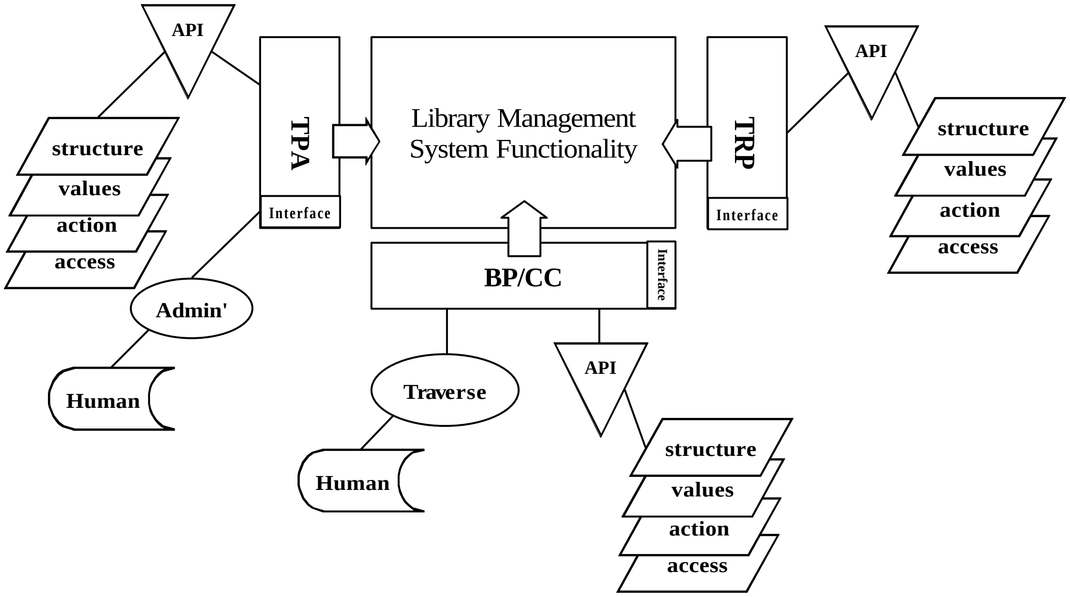
The following matrix attempts to provide a set of ebXML-centric criteria that provide a useful understanding for prioritizing use of middleware solutions.

Factor	WebDav DASL	CORBA	SOAP
Secure interchanges	SSL based	Yes	Yes
http support	Yes	Yes	Yes
Public open standard	Yes	Vendors	Vendors
Database transactional model	Yes	No	No
Query language support	Yes	Extensions	No
Error response model	Yes	Yes	Yes
Access profile support	Yes	Extensions	Yes
Loosely coupled interchange model	Yes	Tight coupled	Semi
Cross-platform support	Yes	Installable	Installable
Apache Web Server extensions	Yes	No	No
XML based syntax	Yes	Support for	Yes
Extensible query/response structures	Yes	Semi	Semi

3.2 Interfacing Models

The ebXML Technical Architecture specifications detail the actual registry/repository interfacing required for each of the components of ebXML. The figure shown here illustrates these as a set of interface services to be provided. This approach allows us to define discrete interface XML structures to implement these with.

Figure 6. The ebXML Registry Interfaces



Shown are three interface components to the major ebXML modules of TRP, TPA and BP/CC. The role and actors (see ebXML Registry/Repository Specifications Part 1) determine the types of interactions supported by these interfaces. Therefore TRP does not warrant a human interface capability since only machine-to-machine interactions are required with the Registry.

The library management system functionality essentially treats the internal mechanisms within the ebXML Registry implementation as a ‘blackbox’ that supports the requirements as laid out in both the overall ebXML Requirements document, the Registry/Repository Part 1 and the Registrar, DocumentManager and TPAManager noted elsewhere in this document. This approach allows any such capable existing document management or library system to be exposed as an ebXML Registry using the appropriate WebDav DASL interfacing bindings.

Each of the interfaces is now described functionally and then in the following section actually interchange XML structure specifications are shown. The common theme is that any registry interface will consist of the components, Access, Action, Structure and Values. These correspond to the similar DASL approach of technology neutral bindings.

The definition of each of these is:

1. Access - The profile that describes the access allowed, includes an optional channel through which information is accessed, and an associated user account and optional password. The user account will have an associated ebXML TPA profile.
2. Action – The particular action to be performed, either a Query, or a Change Request and then an optional post-processing action and optional error action.
3. Structure – the associated XML structure of both the request format and also the response format. These will be associated using either a URL or a namespace.
4. Values – the actual content values in either the request, or the response XML payload details.

3.2.1 The TRP Interface Model

The TRP interface provides a machine level Application Programming Interface (API) using WebDav DASL based interactions. The TRP interface is primarily concerned with verifying transport related content in the ebXML-messaging envelope. For this it requires to access classification structure information, semantic business information and actual content values to ensure compliance. Therefore request/response mechanisms are required for these interactions. The interaction content and functionality themselves are more fully described in the ebXML TRP Specifications.

4.2.1 The TPA Interface Model

The TPA interface provides both a machine level API and a human level interface. The human level interface is required to support TPA management and administration. While API calls will underpin the actual human interface, and the actual mechanics and look and feel of the human interface are not prescribed, it is important to state in the specifications that a human interface is provided. This is to ensure that authentication and verification of critical trading partner information is possible locally for the registry administrator, and other than through a remote API interface. The specific human interface functionality that is required is:

1. The ability to query on and review an individual TPA entry details.
2. The ability to update and change an individual TPA entry details.
3. The ability to setup access profiles and then to assign these to TPA entries.

Meanwhile the API machine-to-machine interfacing provides trading partner information to compliment the TRP API by providing specific verification information and also to provide search capabilities for Business Process related querying. Therefore the TPA API interface may be used to discover capable trading partners within an industry or business process domain. Again, the TRP messaging specifications are sufficiently clear on the requirements to access TPA content and at that level of access require strictly query/response interchanges with optional access logging to implement.

5.2.1 The BP/CC (ebXML GUIDE) Interface Model

The BP/CC interface provides both a machine level API and a human traversal discovery interface. This human interface is intended primarily to be used by business analyst staff researching content and business processes within the registry. Such human interface interactions are intended to use a topic map style presentation of the related information within the Registry organized according to the business process classification system inherent in the Registry. The ebXML GUIDE specifications provide the classification layer content to drive this functionality and the ebXML BP and CC specifications provide the specialized content structures within the classification layer. This functionality is also a discrete focused business tool that allows industry domains to publish their business processes either generically, or particular to either groups of trading partners or individual businesses within the industry. While API calls will underpin the actual human interface, and the actual mechanics and look and feel of the human interface are not prescribed, it is important to state in the specifications that a human interface is provided. Each industry implementation may differ in the style of information presentation and scope made available and this specification is not attempting to dictate those aspects. Instead a list is presented here of human functionality that can be enabled.

1. Tree based topic map traversable structure that provides a review of business domain, and the industry partners and the business processes supported by the registry.
2. Ability to query on a specific classification details within an industry and return a list of applicable element definitions for review.
3. Ability to query on an item by unique reference identifier and return that content item for display and review.
4. The ability to submit changes to the content details within the registry.

The machine API calls that underpin the human interface then provide the same functionality in machine-to-machine interfacing with the BP/CC content within the Registry. By specifying a discrete set of ebXML GUIDE classification structures this reduces the need for ebXML based business applications to perform complex discovery interactions with an ebXML Registry to determine the actual semantics of information content. This both speeds access and makes for more consistently interoperable interchanges.

6.2.1 Alignment with TRP Interface and Security Model

Reviewing the DASL approach and the MIME based approach TRP approach there are significant similarities in the formatting and structure of the interchanges. We do not anticipate that the differences where they exist between the two systems will present particular implementation challenges, particularly as WebDav is now a widely supported open cross-platform specification.

The TRP messaging model already has an envelope structure that contains specific information regarding the trading partner and authentication and verification information. However, these same mechanisms are not always applicable to adopting wholesale for the Registry access, as the business functional needs are different. We also face a very real 'Catch22' situation where the information in the TRP header requires access to the Registry to access the TPA within the Registry. The solution is to link the Registry WebDav DASL accessing to the same content as the TRP exchange uses for TPA verification within the Registry through a lightweight DASL query mechanism that still provides sufficient security and authentication measures. Such information inside the TRP envelope can then be optional encrypted using the recipient's public encryption key. The TRP services can then issue DASL requests based off the information in the TRP envelope header alone and this then ensures consistency.

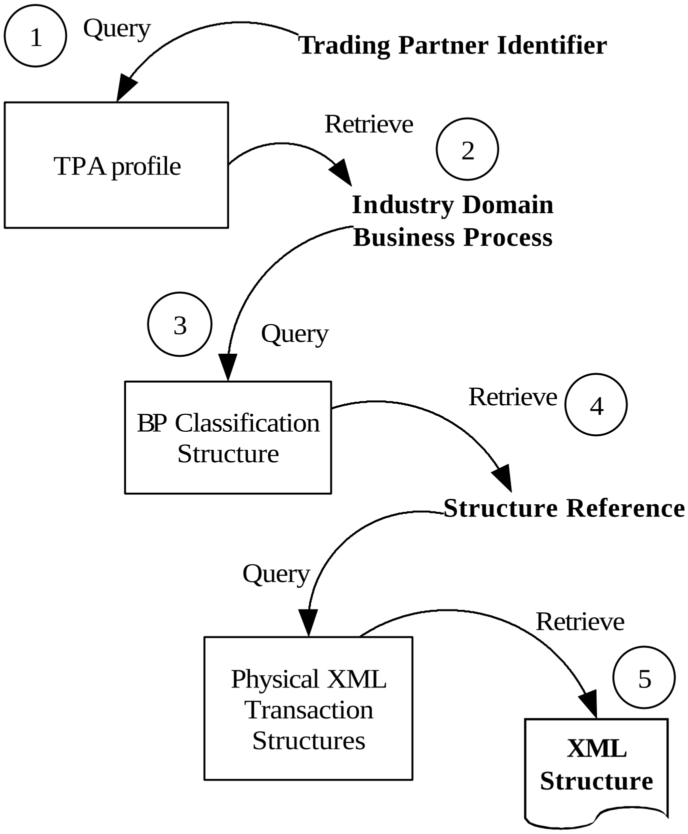
The WebDav DASL system also has its own error response handling system, so this removes the need for ebXML Registry/Repository interfaces to define these mechanisms as they are provided in the interchange.

7.2.1 The Linkage Model between Classification, Interface and Query/Response mechanisms.

To help with the understanding of how the interface mechanisms actually get implemented the following diagram shows how information within the query and response is drawn from the various components of the Registry/Repository itself.

The need is to provide generalized querying mechanisms that are driven off the base primitive structures that are used to define all ebXML BP models, ebXML CC models and reference table implementations. An example of such generic structural based markup is the <definitions> section in the GUIDE element definitions, and the retrieval of EDI igML information using this ability to model any structured information content. See examples 6 & 7 below for this use case. The first set of examples below show a simpler use where the queries retrieve a structure definition based on the BP industry domain (GCI) and the reference QIC code value associated with the structure item itself.

Figure 7. Query/Retrieve Semantic Retrieval Information Interactions.



The next section shows actual syntax examples for this interaction model.

3.3 Examples of Registry Interfacing

The following extracts are provided to aid understanding of this document.

The WebDav DASL approach provides an ideal widely supported lightweight XML based interaction model.

Further more the DASL system provides an extensible interface specification, so ebXML compatible query and response structures can be registered and then utilized within a DASL XML wrapper.   For more information on DASL see http://www.webdav.org).

Example 1 ebXML Registry DASL query structure

This example illustrates a simple query to return a structure content item from the registry. The request below is an implicit XML structure based system that is keyed off the base ebXML classification structures within the ebXML Registry. Since an ebXML Registry is not an arbitrary collection of unordered information, but instead is a focused set of related content the request can utilize basic primitive aspects of the ebXML Registry to enable the request interface system (**Structure Reference** as noted in figure 7 as above).

Therefore the query knows that it can reference the two tags <domain> and <qic> as primitives within a classification structure.   In this example it has already been previously determined by examining the BP classification that the transaction required has a QIC reference identifier of ‘GCI07090’ and is from the industry domain of ‘GCI’.

```
SEARCH / HTTP/1.1
Content-Type: text/xml
Connection: Close
Content-Length: 632

<?xml version="1.0" ?>
 <!-- ebXML Registry Structure Request V0.1 -->
 <D:searchrequest xmlns:D="DAV:" xmlns:eb="ebXML:">
  <eb:request>
   <eb:access>
    <eb:channel>anonymous</eb:channel>
    <eb:auth user="klaus" password="76778jjk" />
   </eb:access>
   <eb:input>
     <eb:match>
        <eb:item name="domain"  value="GCI"/>
        <eb:item name="qic"       value="GCI07090"/>
     </eb:match>
     <eb:select>
         <eb:version>00</eb:version>
         <eb:content>structure</eb:content>
         <eb:parent>root</eb:parent>
   </eb:select>
   <eb:operation>
     <eb:pageSize>10</eb:pageSize>
     <eb:hitCount>1</eb:hitCount>
   </eb:operation>
  </eb:input>
  <eb:output type="content" />
 </eb:request>
</D:searchrequest>
```

Reviewing the request structure above the <eb:match> block contains references to domain and qic items that are part of the ebXML GUIDE classification scheme so therefore these are known structural elements that can be searched on. In fact any element within the registry can be searched on in context using this technique.   DASL also provides the means to specify selection operatives such as <or> and <and> to adjust the search behaviour. By default a <eb:match> block is an implicit logical and of all items specified.   This behaviour will accommodate most common requests to the Registry.

In the <eb:select> block a request for version ‘00’ will return the latest version available, and the content and parent elements indicate that we require the complete structure of the matching XML content.   The <eb:operation> block controls the behaviour of the search process itself. Again DASL provides these mechanisms to control the operation of the search system.

Then the <eb:output> block controls how the output is returned to the invoking system. The “content” parameter causes the default behaviour of returning the physical content, the other option is to return a URL pointer structure that can be used to reference the physical content itself.

Example 2 ebXML Registry DASL response structure

The corresponding response mechanism is now shown for the request query in Example 1 above.

```
HTTP/1.1 207 Multi-Status
Content-Type: text/xml
Content-Length: 2032

<?xml version="1.0" ?>
 <D:multistatus xmlns:D="DAV:" xmlns:eb="ebXML"
xmlns:R="http://www.ebxml.org/dasl-resp-schema">
<D:response>
<D:href />
<D:propstat>
<D:prop>
  <R:author>Ravi Kraft</R:author>
  <R:title>Catalogue Manifest</R:title>
  <R:synopsis>Vendor Catalogue Inventory Details</R:synopsis>
  <R:last-modified>1999-12-25T112222PST</R:last-modified>
  <R:size unit="kilobytes">3</R:size>
  <R:extra-info />
  <R:external-doc-id />
  <R:doc-id>11227726625</R:doc-id>
  </D:prop>
  </D:propstat>
   <eb:structure>
<![CDATA[
<!-- Main definition of CatXML content schema V 1.1 -->
<!ELEMENT Input  (Schema , Content )>
<!ELEMENT Schema (#PCDATA )>
<!ELEMENT Content (Vendor? , Supplier? , StockInfo? , ShipInfo? , Item
)>
<!-- Establish link to qic reference location -->
<!ATTLIST Content
        qicref CDATA #FIXED "http://www.catxml.org/qic/datatypes.xml" >

<!ELEMENT Vendor  (CompanyID , Name? , Address? , Contact? )>
<!ATTLIST Vendor
          vendorID ID #IMPLIED
          qic      'GCI01502' #FIXED >
<!ELEMENT CompanyID  (#PCDATA )>
<!ATTLIST CompanyID
          context (Vendor|Supplier|Manufacturer|Other) 'Vendor'
          idType (DUNS|Local|USDoD|EIN|TaxID|Other) 'DUNS'
          qic      'GCI01503' #FIXED >
<!ELEMENT Name  (#PCDATA)>
<!ENTITY % addressInfo   SYSTEM "CatXML-address-V1.dtd" >
<!ENTITY % contactInfo   SYSTEM "CatXML-contact-V1.dtd" >
<!ENTITY % shippingInfo  SYSTEM "CatXML-shipping-V1.dtd" >
<!ENTITY % usgovDoDInfo  SYSTEM "CatXML-usgovDoD-V1.dtd" >
<!ENTITY % stockInfo     SYSTEM "CatXML-warehouse-V1.dtd" >

```

```
 %addressInfo;
 %contactInfo;
 %shippingInfo;
 %usgovDoDInfo;
 %stockInfo;
  ]]>
  </eb:structure>
 </D:response>
</D:multistatus>
```

The next example shows a return of a link reference to repository content rather than the physical content itself.

Example 3 ebXML Registry DASL response structure

The corresponding response mechanism is now shown for the request query in Example 1 above where the <eb:output> block request is changed to specify a URL instead of the content itself.

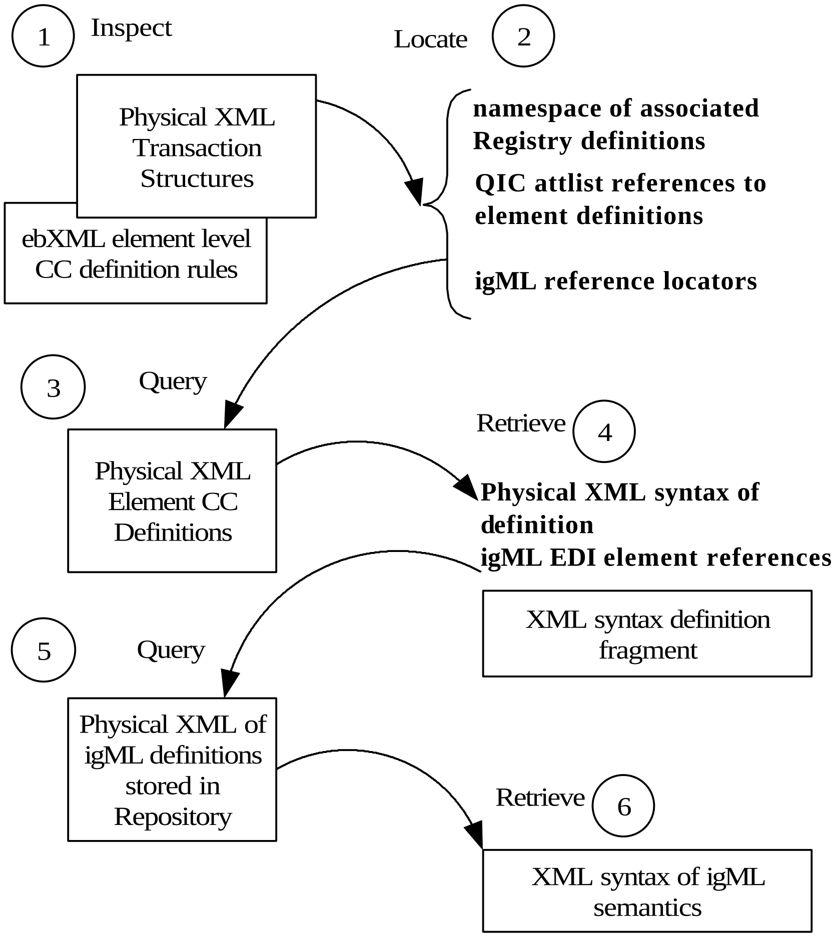
```
HTTP/1.1 207 Multi-Status
Content-Type: text/xml
Content-Length: 763

<?xml version="1.0" ?>
<D:multistatus xmlns:D="DAV:" xmlns:eb="ebXML"
xmlns:R="http://www.ebxml.org/dasl-resp-schema">
  <D:response>
    <D:href>http://www.GCI.org/ebXML/catalogue.xml</D:href>
    <D:propstat>
     <D:prop>
       <R:author>Duane Nickull</R:author>
       <R:title>Catalogue Manifest</R:title>
       <R:synopsis>Vendor Catalogue Inventory Details</R:synopsis>
       <R:last-modified>1999-12-25T112222PST</R:last-modified>
       <R:size unit="kilobytes">12</R:size>
       <R:extra-info />
       <R:external-doc-id />
       <R:doc-id>11227726625</R:doc-id>
      </D:prop>
     </D:propstat>
    </D:response>
   </D:multistatus>
```

The next example illustrates a request for a fragment of content interchange.

Example 4 ebXML Registry DASL fragment query mode

Taking the previous example, the catalogue structure contains references to element items. The definitions of these element items are stored within the registry/repository. The structure itself contains the linkage between the definition and the use in the specific transaction.   The example below shows the use of these embedded references. Given this context information we can then build a query to the registry to retrieve the EDI related information that is contained in the associated igML (see http://www.igML.org) reference XML structure that defines these.

Figure 8. Query/Retrieve of cascading reference to igML EDI semantics.


1
Inspect
Locate
2
Physical XML Transaction Structures
ebXML element level CC definition rules
namespace of associated Registry definitions
QIC attlist references to element definitions
igML reference locators
3
Query
Retrieve
4
Physical XML Element CC Definitions
Physical XML syntax of definition
igML EDI element references
XML syntax definition fragment
5
Query
Physical XML of igML definitions stored in Repository
Retrieve
6
XML syntax of igML semantics

Reviewing the reference structure from Example 2 and relating this to Figure 8, we can see how the cascading reference system works in the actual XML syntax.

The query/response examples shown next then perform the actual retrievals themselves of the interaction items 2, 3 and 4 from Figure 8.

The namespace reference, the Company ID associated QIC reference identifier of ‘GCI01503’ and is from the industry domain of ‘GCI’ are used to create the query.

```
SEARCH / HTTP/1.1
Content-Type: text/xml
Connection: Close
Content-Length: 632

<?xml version="1.0" ?>
 <!-- ebXML Registry Structure Request V0.1 -->
  <D:searchrequest xmlns:D="DAV:" xmlns:eb="ebXML:">
   <eb:request>
    <eb:access>
      <eb:channel>anonymous</eb:channel>
      <eb:auth user="klaus" password="76778jjk" />
    </eb:access>
    <eb:input>
      <eb:match>
          <eb:item name="domain"  value="GCI"/>
          <eb:item name="qicref"
          value=" http://www.catxml.org/qic/datatypes.xml"/>
          <eb:item name="qic"        value="GCI01503"/>
      </eb:match>
      <eb:select>
           <eb:version>00</eb:version>
           <eb:content>fragment</eb:content>
           <eb:parent> GCI01503:igML</eb:parent>
      </eb:select>
      <eb:operation>
          <eb:pageSize>10</eb:pageSize>
          <eb:hitCount>1</eb:hitCount>
      </eb:operation>
    </eb:input>
    <eb:output type="content" />
   </eb:request>
</D:searchrequest>
```

Reviewing the request structure above the <eb:match> block contains references to the items to be used for the query lookup. The qicref item points to the specific registry item to be queried. Notice the repository for this may be a URN that is remotely located and hence the registry will require access to this, or a mirrored copy locally. The <eb:select> block is used in tandem with the <eb:match> block to retrieve just the fragment within the ebXML reference structure that contains the information required.

The next example illustrates both the ebXML reference CC structure for the Company ID item and the response that is return from the fragment query above.

Example 5 ebXML Registry DASL fragment query response structure

The XML content that is actually queried is shown first, and then the resulting response details. The same techniques can then be applied to retrieve the actual igML EDI details that are pointed to by this reference content. (For more details of the igML EDI repository syntax, see the site http://www.igML.org).

Sample Company ID content.

```
<?xml version="1.0" ?>
<!--
* ebXML GUIDE CC element for use with namespace and IDREF    *
* reference system.                                          *
*                                                            *
  -->
<xmlGuide use="element" name="GCI:Catalogues" version="0.1"
   xmlns:datatypes="http://www.ebXML.org/guides/GCI_datatypes.xml"
   xmlns:qic="http://www.ebXML.org/guides/bizcodes.xml">
   <definitions>
    <bizcode qic="GCI01503" qic:base="CompanyID" bizname="companyID">
     <guide>
      <status date="21/02/2000">approved</status>
      <maxlength>15</maxlength>
      <minlength>1</minlength>
      <datatype>string</datatype>
       <mask>U15</mask>
       <values default="">
          <value /> <!-- allowed values can go here when applicable -->
       </values>
       <localdescription xml:lang="EN" xml:space="preserve">The reference
identifier for a company record in a catalogue entry.
       </localdescription>
      <fulldescription xml:lang="EN" mimetype="HTML" >
           http://www.GCI.org/desc/GCI01503.htm</fulldescription>
      <labels>
        <label xml:lang="EN">Company ID</label>
     </labels>
     <seeAlso>
     </seeAlso>
     <dependencies>
      <dependent type="required">GCI01502</dependent>
    </dependencies>
    <attributes>
     <attribute name="context" qic="GCI01570" type="required" />
     <attribute name="idType"  qic="GCI01571" type="required" />
    </attributes>
   </guide>
  <extensions>
```

```
   <extension type="GCI01503:igML"> <!-- This provides EDI mapping -->
     <item type="Format">EDI X12</item>
     <item type="Message">823</item>
     <item type="SegmentRef">N1</item>
     <item type="DictSegment">N1</item>
     <item type="DictDataElement">98</item>
    </extension>
   </extensions>
   </bizcode>

  <!-- More repository definitions of ebXML CC items can go here when applicable -->
   <bizcode qic="GCI01002" qic:base="addrLine" bizname="ADDR:street">
    <guide /> <!-- details go here -->
   </bizcode>
   <bizcode qic="GCI01003" qic:base="cityName" bizname="ADDR:city">
    <guide /> <!-- details go here -->
   </bizcode>
 </definitions>
 </xmlGuide>
```

The corresponding response mechanism is now shown for the request query in Example 4 given previously from the information structure above of the igML extensions information.

```
HTTP/1.1 207 Multi-Status
Content-Type: text/xml
Content-Length: 2032

<?xml version="1.0" ?>
  <D:multistatus xmlns:D="DAV:" xmlns:eb="ebXML"
xmlns:R="http://www.ebxml.org/dasl-resp-schema">
<D:response>
<D:href />
<D:propstat>
<D:prop>
   <R:author>GCI Administrator</R:author>
   <R:title>Catalogue Elements</R:title>
   <R:synopsis>Vendor Catalogue Inventory Details</R:synopsis>
   <R:last-modified>1999-12-25T112222PST</R:last-modified>
   <R:size unit="kilobytes">1</R:size>
   <R:extra-info />
   <R:external-doc-id />
   <R:doc-id>11227726644</R:doc-id>
   </D:prop>
   </D:propstat>
    <eb:structure>
<![CDATA[
    <extension type="GCI01503:igML"> <!-- This provides EDI mapping -->
    <item type="Format">EDI X12</item>
    <item type="Message">823</item>
    <item type="SegmentRef">N1</item>
    <item type="DictSegment">N1</item>
    <item type="DictDataElement">777</item>
   </extension>
  ]]>
  </eb:structure>
 </D:response>
</D:multistatus>
```

The next example illustrates a request for a change of content interchange.

Example 5 ebXML Registry DASL change request structure

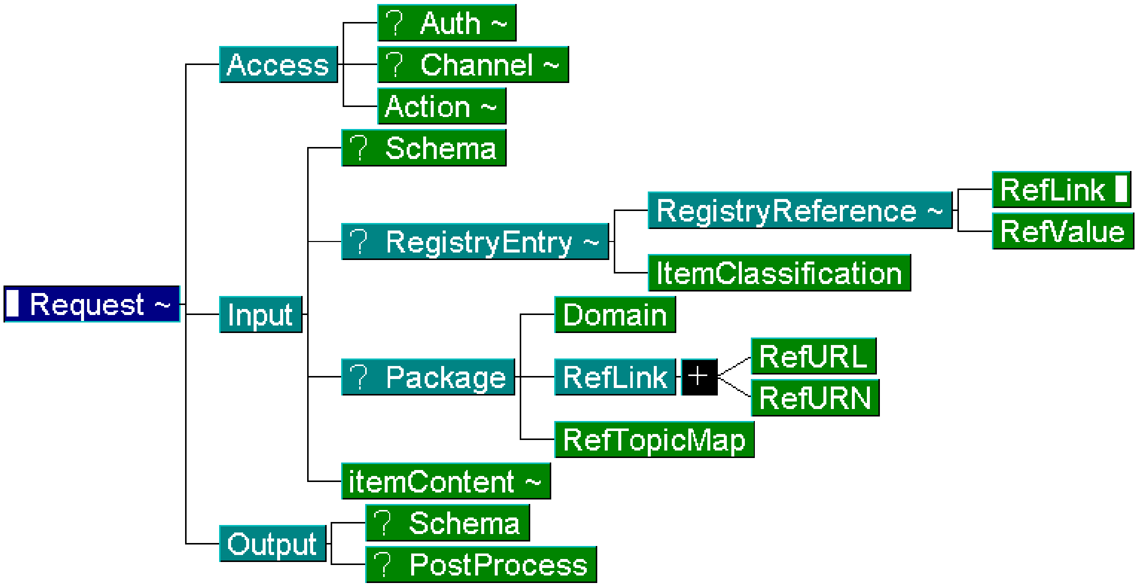
A change request requires more interaction parameters than the simple query. The taxonomy of the ebXML Registry system itself, based on the OASIS and ISO11179 registry functionalities requires that contextual information be associated with the change request to identify the parties concerned, the relation of the content to the registry metamodel, and the status requested for the content itself, and then of course the physical content.

The example below illustrated one such implementation approach. To more fully understand the different interaction semantics the DTD for the update request to the registry must be examined to determine the allowed interactions. The DTD is provided following this example and then in the addendum, along with associated documentation.

```
PROPPATCH /channel/docid#DOC_ID HTTP/1.1
Host: ebXML.company.com
Content-Type: text/xml; charset="utf-8"
Content-Length: xxx
WWW-Authenticate: xxxxxx

<?xml version="1.0" encoding="utf-8" ?>
 <d:propertyupdate xmlns:d="DAV:" xmlns:eb="ebXML:"
xmlns:R="http://www.ebxml.org/dasl-resp-schema">
 <d:set>
<d:prop>
   <R:author>Duane Nickull</R:author>
   <R:synopsis>This is version 2.1 of this address definition</R:synopsis>
   <R:url>http://www.gci.org/ebxml/address.xml</R:url>
  </d:prop>
 <eb:Request lang="EN">
  <Access>
   <Auth userid="scott" passwd="eb7684" session="X25463AS" />
   <Channel name="GCI" code="ALL" />
   <Action verb="Add" noun="Parent" />
  </Access>
  <Input>
   <Schema />
    <RegistryEntry Version="00" ObjectLocation="" DefnSource="ebXML"
      PrimaryClass="defn" SubClass="XML" MimeType="XML"
      ExpiryDate="00-00-0000" ResponsibleOrgURN="www.GCI.org:admin"
      SubmittingOrgURN="xmlglobal:gci" ItemDomain="GCI"
      ItemRegistryURL="http://www.goxml.com/GCI" ItemId="GCI01791">
    <RegistryReference RefDomain="GCI" RefMethod="qic">
    <RefLink>
     <RefURL>http://www.goxml.com/GCI/address.xml</RefURL>
     <RefURN>xmlglobal:gci</RefURN>
    </RefLink>
    <RefValue>GCI01791</RefValue>
   </RegistryReference>
   <ItemClassification>GUIDEstructure</ItemClassification>
  </RegistryEntry>
  <Package />
 <itemContent type="GUIDEstructure" mimetype="XML">
<![CDATA[
<?xml version="1.0" ?>
  <xmlGuide use="structure"
            name="mailingAddress" version="0.1"
      xmlns:qic="http://www.ebXML.org/guides/elements/postal.xml"
      xmlns:crm="http://www.crm.org/guides/elements/basics.xml">
   <sequence>
    <element name="fullName" qic:base="personDetails" />
    <element name="street"  qic:base="postalStreet"
     OCCURS="+" LIMIT="5" />
    <element name="city"    qic:base="postalCity"
                       qic:mask="UX19" />
    <element name="ZIP"     qic:base="usPostalCode" />
    <element name="state"   qic:base="usStateCode" />
    <element name="accountActive"
                  qic:base="crm:activeStatus" />
   </sequence>
  </xmlGuide>
  ]]>
  </itemContent>
  </Input>
  <Output />
  </eb:Request>
  </d:set>
  </d:propertyupdate>
```

The associated DTD for this interaction is thus the following structure. A graphical picture of the compound structure is given first, to aid understanding of the actual mechanisms, and then the physical XML syntax of the DTD itself.

Figure 9. A graphical representation of the Change Request DTD.



Example of the Change Request DTD structure.

```
<!-- ebXML Registry Change Request DTD V0.1 -->
<!ELEMENT Request (Access, Input, Output)>
<!ATTLIST Request
      lang CDATA #IMPLIED
>
<!ELEMENT Access (Auth?, Channel?, Action)>
<!ELEMENT Auth EMPTY>
<!ATTLIST Auth
      userid CDATA #IMPLIED
      passwd CDATA #IMPLIED
      session CDATA #IMPLIED
>
<!ELEMENT Channel EMPTY>
<!ATTLIST Channel
      name CDATA #IMPLIED
      code CDATA #IMPLIED
>
<!ELEMENT Action EMPTY>
<!ATTLIST Action
      verb (Add | Delete | Replace | Supercede | Version) #REQUIRED
      noun (Parent | Fragment | URL | Content) #REQUIRED
>
<!ELEMENT Input (Schema?, RegistryEntry?, Package?, itemContent)>
<!ELEMENT itemContent (#PCDATA)>
<!-- Open element, resolved at runtime -->
<!ATTLIST itemContent
      type (URL | URN | CDATA | MIME | Binary) #REQUIRED
      mimetype CDATA #REQUIRED
```

```
>
<!ELEMENT Output (Schema?, PostProcess?)>
<!ELEMENT Schema (#PCDATA)>
<!ELEMENT PostProcess (#PCDATA)>
<!-- Reference definitions of classification code lists -->
<!ENTITY % assocTypeList "uses | supersedes | contains |  related">
<!ENTITY % contactAvailList "public | priv | prot ">
<!ENTITY % contactRoleList "admin | all | tech">
<!ENTITY % defnSourceList " OASIS | IMS | IEEE_LOM | ebXML | UDDI |
Industry ">
<!ENTITY % stabilityList "comp | dynm | stat">
<!ENTITY % orgRoleList " SO | RO | RA ">
<!ENTITY % primaryClassList "defn | inst | pkg | other">
<!ELEMENT RegistryEntry (RegistryReference, ItemClassification)>
<!ATTLIST RegistryEntry
      Version CDATA #IMPLIED
      ObjectLocation CDATA #REQUIRED
      DefnSource (%defnSourceList;) #REQUIRED
      PrimaryClass (%primaryClassList;) #REQUIRED
      SubClass CDATA #IMPLIED
      MimeType CDATA #REQUIRED
      ExpiryDate CDATA #IMPLIED
      ResponsibleOrgURN CDATA #IMPLIED
      SubmittingOrgURN CDATA #REQUIRED
      ItemDomain CDATA #IMPLIED
      ItemRegistryURL CDATA #REQUIRED
      ItemId ID #IMPLIED
>
<!ELEMENT RegistryReference (RefLink, RefValue)>
<!ATTLIST RegistryReference
      RefDomain (GCI | ebXML | OAG | Other) #REQUIRED
      RefMethod (qic | qicType | mask | IDREF | XLink | XPath | SQL)
#REQUIRED
>
<!ELEMENT RefLink ((RefURL | RefURN)+)>
<!ELEMENT RefURL (#PCDATA)>
<!ELEMENT RefURN (#PCDATA)>
<!ELEMENT RefValue (#PCDATA)>
<!ELEMENT Package (Domain, RefLink, RefTopicMap)>
<!ELEMENT Domain (#PCDATA)>
<!ELEMENT RefTopicMap (#PCDATA)>
<!ELEMENT ItemClassification (#PCDATA)> <!-- reference to
classification -->
```

This DTD makes reference to the classification structure. This is not shown. The classification structure can be an ebXML defined one, such as BP ebXML, CC ebXML or GUIDE ebXML, or can be a user defined classification structure. See the Registry/Repository classification specifications for how to define a classification structure layout.   It is anticipated that Registries will contain sets of pre-defined classification structures for the content they are storing in their repositories to simplify use of the registry and to ensure consistent content and retrievals.

The next section reviews the actual linking mechanisms that support the registry transport layer to resolve URL and URN references within any query/change/response interactions.

3.4 The ebXML RegRep linking

The linking mechanism used in ebXML RegRep is based on either htttp URL links or XML namespaces. The reserved word eb namespace declared in the root tag of the XML transaction instance establishes the reference to the next ebXML RegRep content layer as needed. Therefore a XML transaction will use the eb namespace to reference the structure schema that defines the structural rules, and the eb structure will in turn use its own *element* namespace to locate the default element definitions associated with the structure. The element definitions can also optionally access the *datatypes* namespace to locate datatyping information. This provides an extensible datatype model.

However, fragments that are themselves included, may not have further *include* references within them, thus ensuring that only one level of nesting is provided. Furthermore, permitting only the single ebXML namespace with a single control structure ensures that the true structure of transactions is available and exposed. This contrasts with other early schema implementations that used in-line namespace definitions to retrieve multiple structure schemas, thus creating a system where the true transaction structure could not be determined. The ebXML RegRep avoids this by only allowing the single guide namespace for including the structure linkage.

This linkage mechanism is designed to be simple and business functional and to avoid any complex constructs that make registry implementation and behaviour complex or uncertain. This necessarily restricts the complex use of cascading links, and in particularly linking can only be nested one layer deep, and all recursive references are explicitly not provided.

3.5 Type systems

The ebXML RegRep element definitions use basic business datatypes. All of these are supported by the current W3C datatyping proposal, however the W3C has extended complex behaviours in their datatyping. Any item that does not have a datatype explicitly assigned is treated as a simple string by default.

3.6 Relationship of and use of Bizcodes

The Qualified Indicator Code (QIC) is tied into the Bizcode mechanism that provides the linkage between ebXML classification structures and the associated element definitions and is designed to be a neutral reference code. Use of neutral reference codes is already an established practice within dictionaries of industry element definitions. Therefore many industries already have codes that they can use as QIC references.

The preferred Bizcode QIC structure is a three-letter code, followed by a five-digit number, where the three-letter code denotes the industry or group assigning the codes, and the five-digit number is a sequentially assigned value. It is anticipated that as part of the ebXML repository technical specifications there will also be guidelines established for managing globally unique names under which Bizcode QIC references can be classified.

Currently the barcodes used for product labelling are managed in a similar fashion by having formally registered barcodes alongside locally defined barcodes. With Bizcode QIC labels, since they are tightly coupled to an ebXML classification structure and also stored within an ebXML element repository this already provides excellent separation to avoid conflicts on QIC values assigned within an industry. Also, unlike barcodes where there are many tens of millions already assigned, Bizcodes required a much more limited number since they are reusable across many products. An example is the food industry where there are over seven million barcodes in use, but less than ten thousand unique element definitions (product attributes) are being used to describe all those products.

The current ebXML GUIDE element classification structure is designed to be compatible with ISO11179 based reference registries. The role of ISO11179 registries is to harmonize information classification within a corporation or large government agency for human analytical and business system design purposes. The role of ebXML repositories extends beyond that to include XML based machine-to-machine information interchanges that reference XML repositories via an XML based API and interface specifications.

Therefore ebXML GUIDE classification can be used in tandem with ISO11179, where the ISO registry manages the content that the ebXML system exposes to ebXML aware systems.

4. Tutorial and Use Case

This section presents a short example by the way of an illustration of how to work with and prepare an ebXML RegRep transaction. This section should reference the Tokyo POC implementation documentation.

5. Addendum

A 1. References

W3C Working Draft "XML Schema Part 1: Structures". This is work in progress.

W3C Working Draft "XML Schema Part 2: Datatypes". This is work in progress.

A 1.1 Notes on URI, XML namespaces & schema locations

Namespace use to be defined with regard to the W3C namespace recommendation.

A 1.2 Relative URIs

Throughout this document you see fully qualified URIs used as references. The use of a fully qualified URI is simply to illustrate the referencing concepts.