Open GIS Consortium, Inc.

Date: 2003-01-17

Reference number of this OpenGIS® project document: **OGC 03-028**

Supersedes document: 02-054r1

Version: 0.5

Category: OpenGIS® Interoperability Program Report

Editor: Josh Lieberman (Syncline)
Lou Reich (NASA)
Peter Vretanos (CubeWerx)

OWS1.2 UDDI Experiment

Copyright notice

Warning

This document is not an OGC Standard or Specification. This document presents a discussion of technology issues considered in an Interoperability Initiative of the OGC Interoperability Program. The content of this document is presented to create discussion in the geospatial information industry on this topic; the content of this document is not to be considered an adopted specification of any kind. This document does not represent the official position of the OGC nor of the OGC Technical Committee. It is subject to change without notice and may not be referred to as an OGC Standard or Specification. However, the discussions in this document could very well lead to the definition of an OGC Implementation Specification.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type:	OpenGIS® Interoperability Program Report
Document stage:	Draft
Document language:	English

Contents

i. Preface

This is an OGC Interoperability Program Report for review by OGC members and other interested parties. This document was developed by the UDDI-SOAP Working Group as part of the OGC Interoperability Program OWS 1.2 initiative. The authors of this document are UDDI-SOAP WG members.

ii. Submitting organizations

The following organizations submitted this Implementation Specification to the Open GIS Consortium Inc. as an OpenGIS® Interoperability Program Report:

- Syncline
- NASA
- CubeWerx
- Ionic Software
- Galdos Inc.

iii. Contributors

All questions regarding this submission should be directed to the editor or the submitters:

Josh Lieberman, Syncline Inc. (jlieberman@syncline.com)

Peter Vretanos, Cubewerx (pvretanos@cubewerx.com)

Louis Reich, NASA (lreich@csc.com)

Jerome Sonnet, Ionic Software (jerome.sonnet@ionicsoft.com)

Richard Martell, Galdos (rmartell@galdos.com)

Nadine Alameh, GST (alameh@gst.com)

iv. Revision history

Date	Release	Author	Paragraph modified	Description

v. Changes to the OpenGIS® Abstract Specification

The OpenGIS® Abstract Specification does not require changes to accommodate this OpenGIS® report.

Foreword

Introduction

This document lists the design principles, requirements, and experimental results for future versions of a potential OGC – UDDI (Universal Discovery, Description, and Integration) implementation specification. Specifically, it describes the usage scenarios, workplan, and experimental results for discovery of OGC services (including registries) through the UDDI interface using SOAP (Simple Object Access Protocol) messaging protocols. The baseline for this experiment is the specification for UDDI version 2 and use of private UDDI implementations. Although the draft UDDI Version 3 specification was released in the course of this experiment, software did not become available to make use of it and so it does not receive more than cursory attention in this report. This work was performed during and for the OGC's Interoperability Program OWS1.2 testbed initiative.

The UDDI experiment produced a examples of discovering OGC services through UDDI interfaces, as well as means of mapping between to the UDDI metadata model from metadata models used in OGC services. Work on spatial discovery and content discovery through UDDI showed that these tasks are possible, but that the "fit" with the UDDI model and interfaces is poor at best. This is especially true when considering the capabilities of available UDDI clients to make full (or extended) use of the UDDI service interfaces. The tentative conclusion drawn from this experiment is that UDDI Version 2 is best suited for discovering the existence of services based on very general taxonomic or classification criteria. It is less well suited for obtaining the information to bind to a service, and even less well suited to discovering specific contents or capabilities of individual service instances.

OWS1.2 UDDI Experiment

1. Introduction

This document is a statement of requirements, workplan, and report for an experiment in the use of UDDI (Universal Discovery, Description, and Integration) registries to discover geospatial content in general and OGC services in particular. This work was performed during and for the OGC's Interoperability Program OWS1.2 testbed initiative.

Catalog interfaces and service information models have been developed within OGC specifically for geospatial purposes. This effort has been largely self-contained, however, and not particularly accessible from the Web Services world at large. UDDI, on the other hand, has made the most progress of any service registry towards universal acceptance and accessibility, but has not been specifically adapted for geospatial applications.

The premise of the experiment laid out in this document is to determine whether and how the reach of UDDI might be combined with the geospatial focus of OGC services development to make geospatial content and services more universally discoverable and consumable by non-GIS users. The participants in this experiment will take a variety of approaches to coordinating OGC services and UDDI registries, as expressed in the User Scenarios below. The approaches all center, however, around developing a crosswalk between the OGC and UDDI service information models.

The goal of the experiment will be to assess the practicality of both the crosswalk and the coordination scenarios, as well as to make concrete recommendations for improvements to either or both information models to further this purpose.

2. Relationship to Other Activities

Implementation of UDDI registry interfaces is a part of the general OGC web initiative process. Therefore, the service is tightly related to the following OGC activities:

- Registry Service
- Services Architecture
- Service Information / Service Capabilities

This activity has other relationships, of course, to specification and implementation activities outside of OGC:

- UDDI specification process
- SOAP specification process
- JAXR registry API specification process
- WSDL (Web Services Description Language) specification process

3. Usage Scenarios

There are four general usage scenarios which underlie this experiment. Detailed descriptions of these scenarios are contained in Appendix A of this document.

3.1 Discover OGC Registries

User binds to a general purpose UDDI registry to discover specialized registries (and clients) for geospatial data and services. User then switches to

3.2 Discover OGC Services

User binds to a general-purpose UDDI registry to discover OGC services which have been published to it, either manually or automatically.

3.3 Discover OGC services with UDDI interface to OGC registry

User makes use of general purpose UDDI clients against OGC registries with UDDI interfaces to discover OGC services and build clients to them.

3.4 Publish OGC service to UDDI

User employs a general purpose UDDI publishing client to publish an OGC service directly to UDDI. The service metadata may or may not then be made available through a corresponding OGC registry interface or service.

4. Design Principles

The following design principles should be considered.

4.1 General

The OGC Service Information Model / Registry Model should be mapped onto the UDDI information model with as few changes as possible on either side.

The user scenarios for this experiment will emphasize the differing purposes of UDDI (business and service discovery) and OGC Registry (service and content discovery)

Modifications to either the UDDI or WRS specifications suggested by this experiment will be developed with aim of submission to the respective revision groups for each specification.

4.2 Compatible, Consistent and Extensible

The baseline for this experiment is UDDI v.2.0 , but the effect of changes in UDDI v.3.0 will be considered

4.3 Relationship to other Standards

Compatible with and/or leverages W3C standards efforts such as HTTP and XML.

Compatible with and/or leverages other relevant OGC specification and pre-specifications efforts, including Service Information Model, General Service Model, Registry Information Model, and Web Catalog/Registry Service.

4.4 Accessible and International

OGC use of UDDI registries should be able to conform to the Web accessibility standards (such as those of the W3C Web Accessibility Initiative Content Guidelines).

UDDI clients and service implementations should be able to conform to Web accessibility standards (such as those of the W3C Web Accessibility Initiative Content Guidelines).

All features in the information model mappings developed for this experiment should be available to the international community.

5. Terminology

The key words "**must**", "**should**" and "**may**" are to be interpreted in the detailed requirements as follows:

Must—The item is an absolute requirement of the specification.

Should— There may exist valid reasons in particular circumstances to ignore the item, but the full implications must be understood and carefully weighed before choosing a different source

May—The item will be considered, but further examination is needed to determine if the item should be treated as a requirements.

Note that only the highlighted versions of these terms are to be interpreted as above. Terms that are not highlighted should be interpreted as usual.

.

6. Detailed Requirements

6.1 General requirements

Development and evaluation of a crosswalk between the OGC service information model (SIM) and/or registry information model (RIM) and the UDDI registry information model.

Business information addition to OGC information models

Spatial classification for UDDI RIM

Incorporation of content information into UDDI RIM?

Implementation of OGC service information discovery through a UDDI interface

Implementation of UDDI service information discovery through an OGC interface (optional)

Implementation of compatible WSDL service descriptions and SOAP bindings to allow "automated" discovery and consumption of OGC services through Web Services IDE's (see SOAP experiment DIPR)

Recommendations to UDDI.ORG for revisions to support geospatial services and content offers.

6.2 Proposed UDDI spatial discovery methodologies

Use existing UDDI geographic classification

Use custom quadtree classification (see below)

Use an external tModel validation service to provide spatial searching

Extend existing UDDI interface to support a spatial tModel interface which implements spatial searches.

7. Experiment Proposals

7.1 Cubewerx experiment

The CubeWerx UDDI experiment is a lightweight experiment aimed at gaining some familiarity with using a UDDI registry and thus exploring the capabilities of such registries. The CubeWerx experiment involves the following actions:

Register CubeWerx, as a company, with one or more UDDI registries. CubeWerx will, at a minimum, register with the NASA/Systinet UDDI registry located at:

http://sindbad.gsfc.nasa.gov:8080/uddi/web

and will endeavor to registry with any publicly available, free, UDDI registries such as the IBM or Microsoft registries.

Register all the online services that CubeWerx offers. These include web map services, web feature services, web registry services and web coverage services.

Registry a number of data instances that CubeWerx uses for demonstration purposes.

The registered company, services, and data will be classified using the build-in geographic classification schemes.

One or more new, spatially based, classification schemes (e.g. quadtree LL quadrature) will be created and used to classify the CubeWerx registry entries.

All the registry entries will be made publicly available so that other UDDI/SOAP experiment have something to find and bind to.

The results of the experiments will be reported in this document.

At this time, the CubeWerx UDDI experiment does not include any SOAP based interactions with a UDDI registry.

Note: **If time permits**, the CubeWerx UDDI experiment may include extending the CubeWerx registry so that it automatically updates one or more UDDI (and other) registries whenever a new object is registered or an existing registry object is updated.

7.2 NASA experiment

The NASA experiment is focus on the use of COTS or freeware components within the OWS 1.2 Service discovery framework. It emphasizes the following:

Stand up private commercial UDDI registry implementation for experimental registration of OGC registries and/or other services.

Investigation of interoperability experiments or studies of ebXML Registry/Repositories and UDDI in non-geospatial domains

Investigation of other service discovery workflows involving UDDI related technologies

7.3 Syncline experiment

The Syncline experiment is focused on adding a UDDI interface to an existing OGC registry so that services registered through the OGC interfaces may be discovered by UDDI clients. An optional additional experiment (6) will be to allow publishing of OGC services through a UDDI interface, which then triggers a harvest of that service's capabilities through the OGC registry interface.

Develop crosswalk / mapping from SIM to UDDI

Stand up Sun UDDI registry implementation – DONE

Implement a UDDI discovery interface on an existing OGC service registry.

Try out various methods for spatial discovery of services through UDDI.

Discover OGC services through a UDDI registry using the Visual Studio . NET client.

Implement a UDDI publish interface on an existing OGC service registry. This optional step would probably be accomplished by triggering a GetCapabilities harvest once an OGC service had been published through the UDDI interface.

7.4 Ionic experiment

Use NASA UDDI registry to register their SOAP bindings for WMS and WFS - DONE

Write a client to discover Ionic, Cubewerx, Galdos, etc. services. – August 9

Refine client to publish services to UDDI registry.

User SOAP interface inside Ionic Registry Client to discover services from NASA registry.

7.5 Galdos experiment

Register their OGC registry with UDDI registry (NASA + ?) - DONE

Others can discover Galdos registry and bind to it (using wsdl-tools, etc) - DONE

Map the UDDI v3.0 Inquiry interface to the existing ogcRIM metamodel, thereby turning the Galdos test registry into a UDDI Node. We will take the JAXR mappings as a starting point:

UDDI Inquiry operation	
find_binding	
find_business	
find_relatedBusinesses	
find_Service	
find_tModel	
get_bindingDetail	
get_businessDetail	
get_operationalInfo	
get_serviceDetail	
get_tModelDetail	

This experiment should serve to demonstrate the flexibility of the registry metamodel.

Register the Galdos test registry with a UDDI registry.

8. Experiment Observations

The following sections summarize the current status of the experiments described in the previous section of this document. Due to the voluntary and often unfunded nature of these experiments, many of them are incomplete but some valuable insights into the work performed can nonetheless be gleaned.

8.1 Cubewerx/NASA Observations

8.1.1 Web client experience

One of the components of the Systinet WASP 4.0 UDDI Registry SP1 is a web based client application that implements a GUI front end to the UDDI registry. The client application allows a user to publish business entities, business services, binding templates and tModels.

To register a business, a client must enter the name of the business and an optional description about the business. For each business the client may also enter the services the business offers, contact information for the business, the client may classify the business according the any taxonomy stored in the registry, the client may registry one or more discovery URL's (location of on-line information about the business) and set permission defining who can see the registry record.

Associated with each service is a binding template that is a technical description of how to access the service.

Compatible resources are identified using a tModel – resources with the same tModel are resources of the same type. For the UDDI experiment, tModels were published for the WMS, WFS, WOS, WRS, WCS and the schema of the XML document that Systinet uses to upload taxonomies.

Like the publish case, the web client allows users to query for businesses, services, binding templates or tModels. In each case, two levels of querying are available. A basic level that allows name searches using a variety of query criteria such as SOUNDEX or EXACT MATCH. There is also an advanced query mode that presents a form where all attributes in a record can be constrained.

Using the Systinet web client, CubeWerx was registered as a business and all the OWS1.2 services that CubeWerx offered were published. In addition, several OWS12 layers were registered as tModels. All services and data layers where classified using the built in geographic taxonomies (ISO3166 7& Microsoft Geoweb Taxonomy) as well as the quadcode taxonomy described in section 8.1.2. Queries of the registry using the web client and the integrated clients from Ionic and Laserscan successfully found the registered CubeWerx information.

An attempt was made to register the same information on the IBM and Microsoft public UDDI registries with partial success due to limitations on the type and number of information that could be published. The public registries were not used for any other purpose during the OWS1.2 UDDI experiment.

As reported in section 7.1, CubeWerx did not participate any of the SOAP experiments directly although at least one of the integrated clients used SOAP to access the NASA registry and find the CubeWerx information.

Time and resource limitations did not permit CubeWerx to implement synchronization code between the OWS1.2 service and data registries and the NASA UDDI registry.

8.1.2 Quadcode experiment

8.1.2.1 Introduction

UDDI registries do not natively support spatial queries. In other words, one cannot supply a UDDI registry with a bounding box representing an area of interest and have it identify resource that lie within that region. The purpose of this experiment was to attempt to simulate a spatial coordinate query using the out-of-the-box capabilities of a commercial UDDI registry – specifically the Systinet UDDI registry hosted by NASA.

8.1.2.2 UDDI taxonomies

A standard capability of UDDI registries is the ability to classify entries according to pre-defined or user-defined taxonomies. Systinet UDDI taxonomy management and inquiry is provided by the Taxonomy service and is accessible both through SOAP and the Web client interface. It enables you to browse, create, edit, delete, upload and download taxonomies. Among the available predefined taxonomies, the Systinet registry includes the ISO3166 Geographic Taxonomy and the Microsoft GeoWeb Taxonomy. These can be used to perform spatial queries by place name – "find all records classified as Ontario", for example, will return all records in Ontario. Of course such a spatial query is very limited since you can one locate records attached to a place name and not any arbitrary region of interest on the Earth. However, a UDDI registry's taxonomy management feature can be used to simulate coordinate based spatial queries.

8.1.2.3 Quadcode taxonomy

8.1.2.3.1 Quadcodes

A *quadcode* is a string that is used to represent a precise region in an object space. For example, consider figure 1 where the surface of the earth has been subdivided into four quadrants.

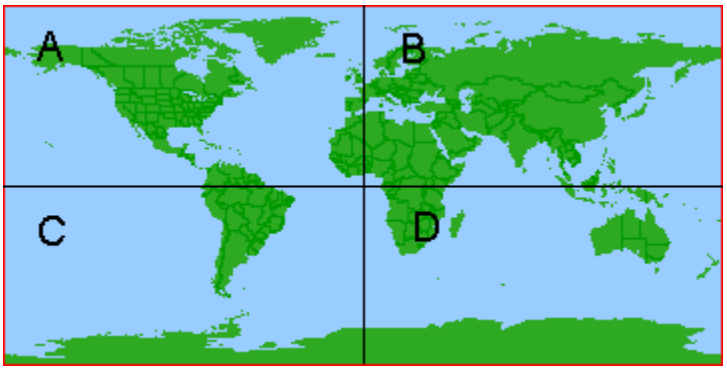


Figure 1.

Each quadrant has been labeled with a letter that is the *quadcode* for that quadrant. Thus the letter ***A*** represents the region from –180° to 0° longitudes and 0° to 90° latitudes. The letter ***B***, from 0° to 180° and 0° to 90°, the letter ***C*** from –180° to 0° and –90° to 0° and the letter ***D*** from 0° to 180° and –90° to 0°.

Quadcodes can be made arbitrarily precise by recursively subdividing the object space until the desired precision is obtained. Consider figure 2, which shows the Earth with 2 levels of subdivision.

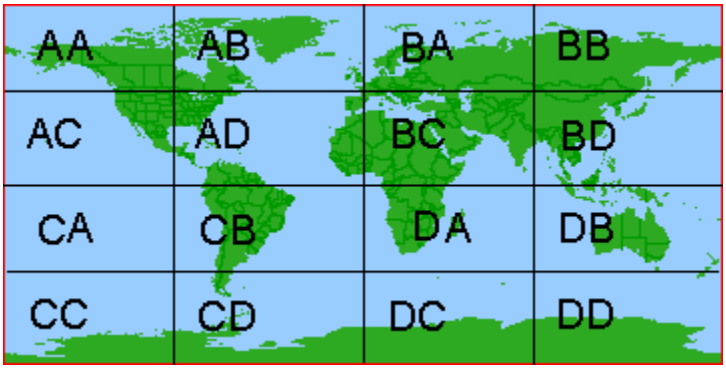


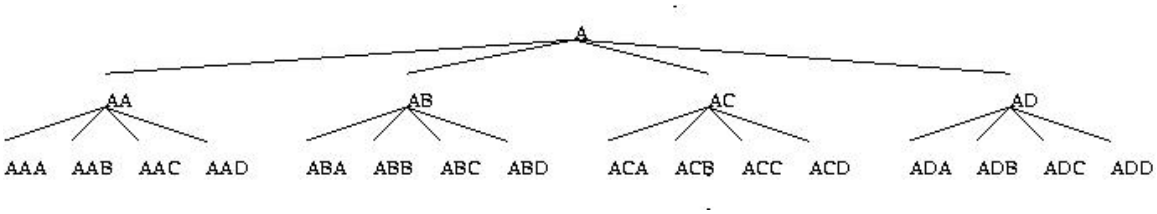
Figure 2.

Each of the original four quadrants in figure 1, have been further subdivided into four child quadrants. Appending the corresponding letter to the parent *quadcode* forms the *quadcode* for each child quadrant. Thus the child quadrants of quadrant ***A*** are ***AA*****,** ***AB*****,** ***AC*** and ***AD*****.** Each child quadrant or cell now represents a correspondingly smaller region on the surface of the Earth. Were this process of recursive subdivision of each child quadrant to continue, there would be $2^{2^{\wedge}level}$ child cells at each level. The length of the quadcode indicates the number of levels of subdivision and thus the precision. In figure 2, there are two levels of subdivision and so the quadcodes are 2 characters long.

Using quadcodes, a hierarchical set of codes that completely cover the Earth's surface can be generated[1]. Figure 3 illustrates such a hierarchical set of quadcodes as a tree for parent quadrant A:

[1] At each level, the quadcodes represent a non-overlapping tessellation of the Earth's surface.

Figure 3.



Similar trees can represent the other quadrants.

8.1.2.3.2 Taxonomy key values

In a UDDI registry, a hierarchy like that illustrated in figure 3 can be represented as a user-defined taxonomy and used to classify information in the registry. During the UDDI experiment, the term *quadcode taxonomy* was used to refer to a taxonomy composed of quadcodes. Each quadcode in the hierarchy represent a key value in the quadcode taxonomy.

Registry entries would be classified using the leaf nodes of the quadcode taxonomy. For example, figure 4 shows the quadrants around Toronto, Canada to 5 levels of subdivision.

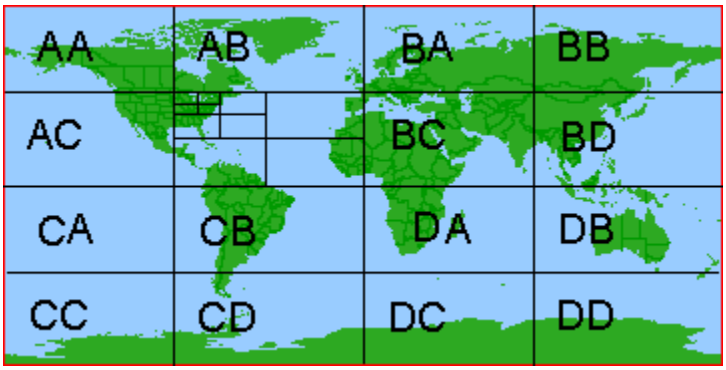


Figure 4.

According to figure 4, a business in Toronto could be classified using the key value ADAAD to indicate the quadrant in which the business is located. Thus any UDDI taxonomy query on quadrant A, AD, ADA, ADAA or ADAAD would find Toronto.

8.1.2.3.3 Loading a quadcode taxonomy

The Systinet UDDI registry taxonomy management service includes the ability to bulk load (or *upload* in Systinet lingo) a taxonomy encoded as an XML document. Systinet does not provide a schema for the XML document, but does include several examples that illustrate the structure of the taxonomy encoding document.

For the UDDI quadcode experiment, a program called *QuadTaxonomy* was written that generates an XML document containing a quadcode taxonomy to a specified level of subdivision. The following is the output that *QuadTaxonomy* generates for 1 level of subdivision (the line numbers are for reference only and are not actually generated by the program):

```
1.  <?xml version="1.0"?>
2.  <!-- <!DOCTYPE taxonomy SYSTEM "taxonomy.dtd"> -->
3.  <taxonomy>
4.     <tModelKey>uuid:65bee600-f1c1-11d6-b493-b8a03c50a862</tModelKey>
5.     <name>OGC Quads</name>
6.     <checked>true</checked>
7.     <compatibility>
8.        <type>businessEntity</type>
9.        <type>businessService</type>
10.    </compatibility>
11.    <categorization>
12.       <type>categorization</type>
13.    </categorization>
14.    <categories>
15.       <category>
16.          <keyName>(-85,36.5) to (-76.5,45)</keyName>
17.          <keyValue>A</keyValue>
18.       </category>
19.       <category>
20.          <keyName>(-76.5,36.5) to (-68,45)</keyName>
21.          <keyValue>B</keyValue>
22.       </category>
23.       <category>
24.          <keyName>(-85,28) to (-76.5,36.5)</keyName>
25.          <keyValue>C</keyValue>
26.       </category>
27.       <category>
28.          <keyName>(-76.5,28) to (-68,36.5)</keyName>
29.          <keyValue>D</keyValue>
30.       </category>
31.    </categories>
32. </taxonomy>
```

The *taxonomy.dtd* file referenced in line 2 is only a placeholder for an actual schema file.

The tModel key specified in line 4 is a reference to a model that simply points to an empty *taxonomy.dtd* file at http://www.pvretano.com/uddi/taxonomy.dtd.

The names generated for each category are encoded using the *keyName* element and represent the extent in the object space of each quadcode.

The original UDDI experiment generated a quadcode taxonomy for the entire world to 10 levels of subdivision. This, however, resulted in a 350 megabyte XML file which the Systinet installation at NASA was unable to upload. In fact, the upload procedure corrupted the underlying Postgres database system and a recovery procedure was required to restore the system.

In order to make the XML file smaller, a new taxonomy file was generated that restricted the extent of the object space from the entire world ([-180,180][-90,90]) to the east coast of North America ([-86, -68][28, 45]) and only generated 8 levels of subdivision. This resulted in a 19 megabyte XML file that the NASA registry could easily handle.

8.1.2.4 Administering the Taxonomy

Version 4.0 features limited taxonomy management. It allows only administrator to upload a taxonomy. We had fairly large taxonomy to upload (20MB); however, it still took unreasonable amount of time to do that. On average it was taking 24 hours to upload the taxonomy on the specified above machine. According to Systinet, in version 4.5 of the registry, the taxonomy upload mechanism was completely rewritten with significant performance improvements. Even though the web interface allows the administrator to delete taxonomies, if taxonomy upload partially failed (as happened to us several times), it is impossible to delete the taxonomy using the interface. The only way to delete the taxonomy then was to go directly to the database's interface and delete all the pieces of taxonomy via SQL statements. Once taxonomy is uploaded, it is not automatically visible to anyone but administrator and only via administration web interface. While the registry features very extensive documentation, we were not able to find the way how to make the taxonomy visible to everyone. Instead, we had to turn to customer support and discussion groups where we found this information.

.

8.1.2.5 Spatial Queries

Section 8.1.2.3, describes how quadcodes can be generated and how they can be loaded into a Systinet UDDI registry as a taxonomy. This section describes how such a taxonomy can be used to perform spatial queries.

Consider a portion of figure 4, reproduced here as figure 5.

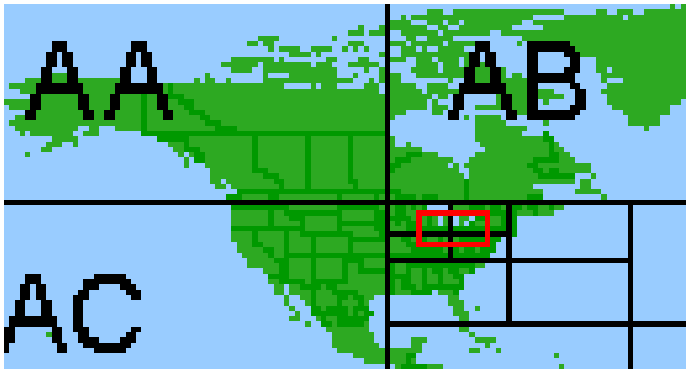


Figure 5.

The red box represents a spatial query window or area of interest. The box overlaps the quadcodes ADAAA, ADAAB ADAAC and ADAAD. These quadcodes are also key values in the quad taxonomy. So, if a taxonomy query is performed on a UDDI registry to find all records classified with the key values ADAAA, ADAAB, ADAAC and ADAAD then we have in essence performed a spatial query. Of course, this is an *approximate* spatial query and represents a first pass filter[2]. Further processing of the results at the client would be required to perform an exact spatial query.

8.1.2.6 Tessellation service – converting BBOX to quadcodes

In section 8.1.2.4 it was explained that a spatial query using quadcodes is a simply a matter of finding which quadcodes the query window overlaps and then using those quadcodes as key values in a taxonomy query on the UDDI registry.

Since the coordinates of each quadcode are easily calculated based on the extent of the object space (see section 8.1.2.3.1), it is a simple matter to convert the coordinates of a bounding box into a list of overlapping quadcodes. The process of performing this conversion of a bounding box to quadcodes is called *tessellation* and there are numerous references describing efficient tessellation algorithms[3].

---

[2] The procedure of identifying overlapping quadcodes is known as a *spatial join*. In essence a join is being performed between the list of quadcodes that represent the BBOX (ADAAA, ADAAB, ADAAC and ADAAD) and the list of key values in the quadcode taxonomy to see if there are any common or overlapping cells.

[3] Samet, Hanan, The Design and Analysis of Spatial Data Structures, Addison Wesley, 1989

For this UDDI experiment a tessellation web service was implemented to convert a bounding box in a specified spatial reference system into a list of quadcodes (i.e. key values) that can be used on the NASA Systinet registry to perform a taxonomy query. The existence of such a web based service means that registry clients do no have to implement the tessellation algorithm themselves and need only parse the XML output to` generate an appropriate UDDI taxonomy query.

The following example illustrates how to invoke the tessellation service for the UDDI experiment:

http://www.pvretano.com/cwquad/cwquad.cgi?BBOX=-78.0,38.7,-77.9,38.9

The output for this request is the following XML document:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<categories xmlns="http://www.opengis.net/ows12_uddi"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/ows12_uddi
http://schemas.cubewerx.com/schemas/uddi/OWS12_UDDI_Categories.xsd">
    <category>
        <keyName>(-84.0039,44.0039) to (-83.9375,44.0703)</keyName>
        <keyValue>AAAADDDB</keyValue>
    </category>
    <category>
        <keyName>(-84.0039,44.0703) to (-83.9375,44.1367)</keyName>
        <keyValue>AAAADDBD</keyValue>
    </category>
    <category>
        <keyName>(-83.9375,44.0039) to (-83.8711,44.0703)</keyName>
        <keyValue>AAABCCCA</keyValue>
    </category>
    <category>
        <keyName>(-83.9375,44.0703) to (-83.8711,44.1367)</keyName>
        <keyValue>AAABCCAC</keyValue>
    </category>
</categories>
```

The parameters of the web tessellation service are specified in the following table.

URL Component	O/M	DEFAULT	Description
BBOX	M		The coordinates of the BBOX specified as lower-left point and upper-right point.
SRS	O	EPSG:4326	The SRS in which the BBOX is specified.

MAXLEVELS	O	8	The maximum number of levels to which to generate the quadcodes (i.e. taxonomy key values).

Table 1. – Web Tessellation Service Parameters

8.1.2.7 Client API

The goal of the UDDI quadcode experiment is to be able to perform coordinate based spatial queries on a UDDI registry using only the features supported by the registry – namely the ability to perform taxonomy queries. Any client application wishing to perform a spatial query would need to perform the following steps described in sections 8.1.2.3, 8.1.2.4 and 8.1.2.5:

1. Invoke the web based tessellation service to generate a set of quadcodes that overlap the query window.

2. This set of quadcodes represents key values that can be used to execute a taxonomy query on a UDDI registry to identify records that lie within the query window.

In order to make it easier to perform spatial queries on a UDDI registry, a Java class was written that performs the necessary steps and can be integrated into a client application. Due to time constaints of the client providers NASA wrote athe JAVA class.This section describes the experience of writing the client API.

The following third party libraries were used to implement the client API.

- UDDI4J – needed to programmatically interact with UDDI registry
- JDOM – XML DOM Java wrapper library
- Xerces – XML Parser

UDDI4J was chosen over Systinet's UDDI client library in order to test the level of interoperability that could be achieved between the UDDI registry and a UDDI client library produced by different vendor.

Internally, the API consists of three logical parts:

1. Code needed to perform call to the CGI-based tessellation service

2. Code needed to performs calls to UDDI registry to perform the taxonomy query.

3. Support code to handle command line arguments and data received from UDDI registry and CGI-based service.

The implementation of the API was very straightforward. Using existing **java.net** classes the code makes a call to the web based tessellation service. Then using JDOM and Xerces the returned XML document, containing the quadcode taxonomy key values, is parsed and converted into a UDDI *CategoryBag*. The *CategoryBag* is then used to search for *ServiceInfos*. The found *ServiceInfos* are in turn used to search for binding details for the found services. Finally, each service binding detail is converted into a HashMap where the key is the service name and the access point is the value.

While implementing the client the following observations where made:

1. *FindQualifiers* object can be used to control how parameters in a *Bag* (e.g. *CategoryBag*) are passed to the call (logical AND, OR etc.)

2. Initially the tessellation service was returning key values that did not match any quadcode classifications in the UDDI registry. As a result, the client code was unable to find any result. Once this problem was rectified, the client code worked perfectly.

Both Ionic and LaserScan used the client API in their integrated OWS1.2 clients and were able to successfully perform spatial queries using bounding boxes on the NASA UDDI registry.

This provided a successful conclusion to the interoperability test between UDDI4J client library the Systinet UDDI server.

The source code for the java class can be found in Annex C.

8.2NASA Observations

8.2.1 Installing and Maintaining a Commercial UDDI Registry

This section describes administrative experience of Systinet WASP UDDI registry version 4.0. It briefly outlines pluses and minuses of various aspects of administration and running the registry. All the work was performed on Dell PC with Pentium III 667MHz [processor with 512 MB of RAM, running RedHat Linux 7.3.

8.2.1.1 Installation/Upgrade

Systinet WASP UDDI 4.0 for Linux features easy to use command line installation/upgrade script. It guides the administrator through the process by asking to input serious of values (with defaults given). In case of upgrade, it asks to enter the

location of previous installation so it then copies new binaries in that directory while keeping all the old settings.

8.2.1.2 Configuration

After installation is complete, the administrator needs to start the registry with the provided script. Then, the one needs to go to the provided web interface to modify some key values(e.g.name of the database to use and jdbc driver name, administrative email contact information). Overall, there are number of configuration parameters that can be modified to customize the registry operation. They also can be left as is. The registry configuration interface is easy and intuitive to use.

8.2.1.3 User administration

Administrator doesn't have to do much to administer users. When new users want to use the registry, they are directed to the appropriate web interface where they register. Once registered, an email with activation URL is sent to the specified email address.

Administrator can then if needed to add users to ACL groups if needed, or change their access level as needed..

8.2.1.4 ACL management

The administration interface features extensive ACL management functionality. It allows administrator to create ACL groups, add users to them. Administrator can change access level to items in the registry (e.g business, service) either by given certain permissions to whole group or individual users.

8.2.1.5 Logging

The registry doesn't have dedicated logging mechanism . While it is possible to redirect stdout and stdrr output to a file, it would be preferable to have logging with time and data markings.

8.2.1.6 Conclusion

Overall, Systinet WASP UDDI Registry Version 4.0 is easy to use and administer. It features simple but sophisticated administration interface. Allowing for quick initial setup and ongoing administration.

8.2.2 Web Service Inspection Language

Web service information discovery is characterized by one or more WSDL or other description documents being used to provide "contact" (that is, endpoint) information in lieu of telephone numbers or physical addresses. The following two sections provide a brief description of how the UDDI and WS-Inspection mechanisms fit into a taxonomy of discovery mechanisms in terms of the characteristics which they exhibit. The final section describes how the two may be applied, depending upon the desired functionality and operating limitations

8.2.2.1 UDDI characteristics

UDDI implements service discovery using a centralized model of one or more repositories containing information on multiple business entities and the services they provide. You could compare UDDI to the Yellow Pages in your phone book, where multiple businesses are grouped and listed with a description of the goods or services they offer and how to contact them.

UDDI systems are based upon organized repositories which provide many high-level functions, including advanced searching capabilities. This makes them very adept in facilitating focused discovery patterns, including helping requesters quickly locate potential communication partners. To a lesser extent, UDDI is also able to facilitate some patterns of unfocused discovery through browsing of the repository. In order to provide advanced functionality, however, UDDI requires that a certain amount of infrastructure be deployed and maintained, thus increasing the cost of operation. The specification provides a high level of functionality through Simple Object Access Protocol (SOAP) by requiring specifically an infrastructure to be deployed with substantial overhead and costs associated to its use.

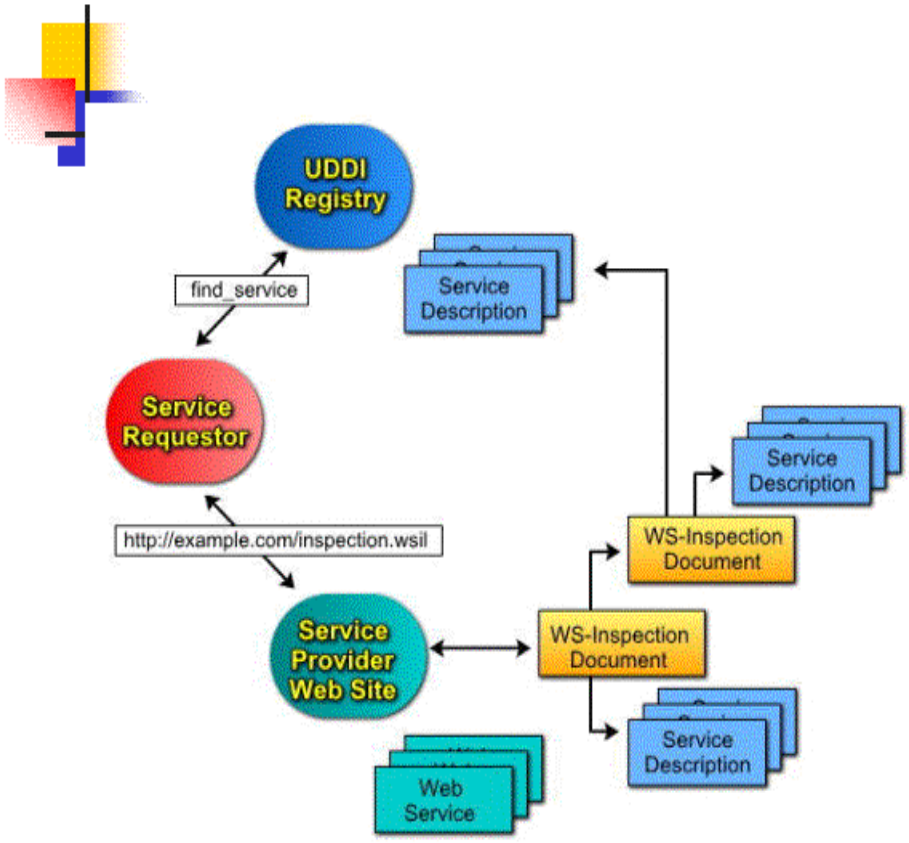
8.2.2.2 WS-Inspection characteristics

The Web Services Inspection Language (WS-Inspection) [3] relies upon a completely distributed model for providing service-related information; the service description information can be distributed to any location using a simple extensible XML document format. Unlike UDDI, it does not concern itself with business entity information, nor does it specify a particular service description format. WSIL works under the assumption that you are already familiar with the service provider, and relies on other service description mechanisms such as the Web Services Description Language (WSDL).

WSIL documents are located using a few simple conventions using existing Web infrastructure. In many ways, WSIL is like a business card -- it represents a specific entity, its services, and contact info, and is typically delivered directly by whom it represents. The low functionality and lightweight nature of WSIL leaves the processing for the developer to implement. WSIL begins to break down and become too unwieldy to search or manage if a given document grows too large, or a collection of documents aggregates too deep. Eventually, development effort will diminish as WSIL toolkits are developed, like the one found in the Apache Software Foundation's Axis project The low functionality of a simple XML document format also provides the flexibility for novel and innovative applications of this information to be easily created.

By providing the ability to disseminate service related information through existing protocols directly from the point at which the service is being offered, the WS-Inspection mechanism enables focused discovery to be performed on a single target. Due to its decentralized nature however, the WS-Inspection specification does not provide for a good mechanism upon which to execute focused discovery if the communication partner is unknown. Unlike with business cards and pamphlets, the WS-Inspection specification supports a significant number of unfocused discovery patterns by providing a set of conventions to make its documents easily locatable and to allow them to be presented in a proactive fashion by the service provider. As is the case with other simple aggregation supporting discovery mechanisms, there is a small cost associated with creating and maintaining the aggregations. The WSIL model is more RESTful then UDDI. In many ways, WSIL is like RDF Site Summary (RSS) for Web services. RSS is a file format with pointers to published content that can be syndicated and aggregated. WSIL is a file format with references to published Web services that can be discovered and bound.

As you can see, while UDDI and WSIL are both mechanisms for Web services discovery, their models are quite different. Deciding which you should use depends on your situation. In many cases, it may be advantageous to use both. . Figure 1 provides an overview of the relationship of UDDI and WSIL



As you can see, WSIL can be used to "point" to UDDI repositories and the service descriptions therein.

8.2.2.3 Locating WSIL Documents

Once an inspection document has been created, a consumer needs to be able to find it. WSIL's decentralized document-based model would make locating these files difficult if it were not for a couple of simple conventions defined in the specification.

The first convention employs the use of a fixed filename of inspection.wsil located in common entry points. Consumers would only need to ping a URL such as *http://example.org/inspection.wsil* or *http://examples.org/services/inspection.wsil* to discover the file's existence for retrieval

The second convention employs embedded references in other documents, such as HTML or other WSIL documents. WSIL advocates use of a meta tag in an HTML document that establishes a link to the inspection documentation location.

8.2.3 UDDI Registry V3 and ebXML Reg/Rep v2.3 Information Model and Functionality as Basis of a General Purpose Application Registry

This section will summarize work done by a team of Boeing Architects and posted to the OAISIS UDDI and ebXML RegRep mailing lists. The most recent summary of this work is presented as Annex B of this document. As this section matures this material will be augmented and refined.

8.2.3.1 Web Service Inspection Language

8.2.3.1.1 **Web services discovery**

Web service information discovery is characterized by one or more WSDL or other description documents being used to provide "contact" (that is, endpoint) information in lieu of telephone numbers or physical addresses. The following two sections provide a brief description of how the UDDI and WS-Inspection mechanisms fit into the taxonomy described in the introduction in terms of the characteristics which they exhibit. The final section describes how the two may be applied, depending upon the desired functionality and operating limitations.

8.2.3.1.2 UDDI characteristics

The Universal Description Discovery and Integration (UDDI) specification [2] addresses the problems associated with Web service discovery through the use of a centralized model; one or more repositories are created to house information about businesses and the services which they offer. Requests and updates pertaining to the service and business related information are issued directly against the repositories. In addition, UDDI prescribes a specific format for a portion of the stored description information and, to facilitate advanced searching, assumes that other description information will be stored/registered within the system as well.

In terms of the personal information discovery context which was described above, the UDDI environment most closely resembles that of a directory assistance provider or searchable on-line "Yellow Pages" system. Like directory assistance and other third-party providers, UDDI systems are based upon organized repositories which provide many

high-level functions, including advanced searching capabilities. This makes them very adept in facilitating focused discovery patterns, including helping requesters quickly locate potential communication partners. To a lesser extent, UDDI is also able to facilitate some patterns of unfocused discovery through browsing of the repository. In order to provide advanced functionality, however, UDDI requires that a certain amount of infrastructure be deployed and maintained, thus increasing the cost of operation. In addition, unless the service descriptions are stored only within UDDI, there is a cost associated with keeping the different versions synchronized. Depending upon the business model adopted by the directory provider, these additional costs may or may not directly impact the owner of the information. For instance, the model adopted by the Universal Business Registry shields the information owner from these costs.

8.2.3.1.3 WS-Inspection characteristics

The Web Services Inspection Language (WS-Inspection) [3] relies upon a completely distributed model for providing service-related information; the service descriptions may be stored at any location, and requests to retrieve the information are generally made directly to the entities which are offering the services. The WS-Inspection specification does not stipulate any particular format for the service information; it relies upon other standards, including UDDI, to define the description formats. The WS-Inspection specification also relies upon existing Web technologies and infrastructure to provide mechanisms for publishing and retrieving its documents.

In terms of the personal information discovery context described in the introduction, the WS-Inspection mechanism most closely resembles business cards and other simple information aggregation documents. As is the case with those other mechanisms, WS-Inspection documents are very light-weight, easy to construct, and easy to maintain. By providing the ability to disseminate service related information through existing protocols directly from the point at which the service is being offered, the WS-Inspection mechanism enables focused discovery to be performed on a single target. Due to its decentralized nature however, the WS-Inspection specification does not provide for a good mechanism upon which to execute focused discovery if the communication partner is unknown. Unlike with business cards and pamphlets, the WS-Inspection specification supports a significant number of unfocused discovery patterns by providing a set of conventions to make its documents easily locatable and to allow them to be presented in a proactive fashion by the service provider. As is the case with other simple aggregation supporting discovery mechanisms, there is a small cost associated with creating and maintaining the aggregations.

8.2.3.1.4 The complete picture

Before the picture can be completed, a third mechanism which is analogous to verbal communication from the personal information discovery space must be mentioned. This mechanism involves the direct retrieval of description documents, WSDL and other related files, from their source. As is the case with the verbal scenario, this mechanism does not really have any overhead associated with it, but it does only support portions of the focused and unfocused discovery patterns.

In summary, the Web service discovery mechanisms that have been described possess the following characteristics:

Direct description retrieval (voice, FTP, HTTP GET)

- Supports some focused and unfocused discovery patterns.
- Dissemination is direct from the source/originator.
- No overhead.

Simple aggregate publishing (WS-Inspection)

- Supports some focused discovery patterns and a significant number of unfocused ones.
- Dissemination is direct from the source/originator.
- Moderate overhead.

Advanced directory (UDDI)

- Supports a significant number of focused discovery patterns and a some unfocused ones.
- Dissemination is via a third-party.
- Moderate overhead for the information owner; high overhead for the directory provider.

Like the business cards and the directory assistance systems, the UDDI and WS-Inspection specifications address different sets of issues in the discovery problem space, and are characterized by different sets of trade-offs. UDDI provides a high degree of functionality, but it comes at a cost of increased complexity. The WS-Inspection specification adopts a lower level of functionality in order to maintain a low overhead. In this light, the two specifications should be viewed as complementary technologies, to be used either together or separately depending upon the situation. For example, a UDDI repository could be populated based upon the results found when performing a "Web crawl" for WS-Inspection documents. Likewise, a UDDI repository may itself be

discovered when a requester retrieves a WS-Inspection document which references an entry in the repository. In environments where the advanced functionality afforded by UDDI is not required and where constraints do not allow for its deployment, the WS-Inspection mechanism may provide all of the capabilities which are needed. In situations where data needs to be centrally managed, a UDDI solution alone may provide the best fit. The UDDI and WS-Inspection specifications should not be viewed as providing competing mechanisms, any more than business cards and directory assistance services are viewed as competing to disseminate personal information.

8.2.4 UDDI taxonomy for ebXML technologies

This facilitates publishing ebXML-based services in the UDDI Business Registry (UBR) such that they can be explicitly sought or, if found incidentally, formally identified. This section will be based on a draft technical note in the OASIS UDDI technical committee.

8.3 Syncline Observations

8.3.1 SIM - UDDI mappings

It came as no surprise that some elements of SIM (Service Information Model) and UDDI v. 2 are quite similar and others are vastly different.

UDDI Entity	SIM Entity	Notes
businessEntity	Iso19119:OrganizationName	Not 1st class in SIM
businessService	ServiceInstance	
bindingTemplate	ServiceInstance	Overlaps into ServiceType
tModel	ServiceType	This is only one use of tModel in UDDI, and doesn't allow referencing interface WSDL
keyedReference	ContentInstance	Specific unvalidated usage of general tModel "bag" in UDDI
keyedReference	ContentType	"

keyedReference	PresentationScheme	“

The only real coincidence between SIM and UDDI comes in the area of ServiceInstances and ServiceTypes. A businessService corresponds more-or-less to a ServiceInstance. There is presently no equivalent object to businessEntity in SIM. Although one use of tModel is to represent the equivalent of a service type, it doesn’t provide a place to reference a type definition (e.g. interface WSDL). Instead, a definition reference such as a WSDL document URL can be made in a tModelInstance, which associates a particular businessService with a tModel. There is no direct equivalent for the other SIM high-level information objects.

8.3.2 Use of category and identifier tmodels for SIM

The key (!?) to discovery in UDDI is the object referred to as a tModel. Unfortunately this structure is used in several widely divergent ways in UDDI, sometimes referring to types and sometimes to instances, sometimes representing a descriptive or identifying taxonomy and other times an element itself of a taxonomy. The UDDI interfaces permit searching for tModels themselves, or for other objects (e.g. businessServices) according to keyedReferences, which are simply flat name-value pairs associated to a given tModel.

Although the term taxonomy is used for sets of these key-value pairs, it is up to a given UDDI registry implementation whether a controlled vocabulary of those pairs is maintained (validation); there is no standard interface for maintaining such vocabularies. There is also no explicit organization of the pairs. They are searched in textual order, so a hierarchical search can be made only if the actual contents (value field) are organized hierarchically. The name parameter is not even considered (in the spec) except if the tModel is “uddi-org:general_keywords”.

The implication of this arrangement is that the UDDI model could be “abused” to swallow any desired SIM information by casting it as name-value pairs where the names represent the xpaths to each SIM element/attribute to be included and the values represent the contents of that element/attribute (un-validated). One could even register as descriptors the values of the OGCBasic query parameters, with the understanding that only entire service instances (with contents as these name-value pairs) or tModels (without definitions) could be returned by the UDDI registry. One could even represent the spatial location represented by a service as one hierarchical latitude name-value pair and one longitude name-value pair (so that the number of digits represented the level of spatial resolution, e.g. 10 digits), then AND the pairs in a find_service query which used “like” to find the set of all services within a larger area (e.g. 7 digits). This is obviously not a query for which any existent UDDI registry is likely to be optimized, however.

Another problem with this approach is that, while it could be implemented, it would fail the goal of allowing standard UDDI clients to access the information. It would be unlikely that they would be able to present the "large" contents of a service "categorybag" in any sensible way, or provide a graphical spatial search capability to make use of the lat-lon location pairs.

8.3.3 Use of external taxonomy validation

The role of an external taxonomy validation service in the UDDI v.2 specification turns out to be limited to validation of individual name-value pairs within a tModel "namespace". There is no interface for either generating or searching among such pairs externally. This functionality is therefore strictly useful for maintaining a controlled vocabulary, but not for increasing something such as the searching functionality or assisting a client to use a controlled vocabulary, particularly a dynamic one such as a spatial identifier taxonomy.

8.3.4 Results of testing with Sun UDDI registry

An instance of the UDDI registry provided by Sun in their Web Services Developer Pack was implemented for use in the experiment. This registry is based on a Xindice XML database, as opposed to the relational database orientation of Syncline's OGC registry. We were able to register businesses and services in a nominal fashion to this registry, but the supplied client was inadequate to make use of any more of the publisher API than those operations. Lack of resources and prevented us from doing custom development to pursue either automated synchronization between the UDDI and OGC registries, or the UDDI registry "abuse" described above, when it became clear that there would be little benefit to proving whether this was or was not practical to implement.

8.4 IONIC Observations

TBD.

8.5 Galdos Observations (UDDI to ebRIM mappings)

8.5.1 Introduction

The registry provides experimental functionality to allow the client to query for content using UDDI inquiries. A two-stage mapping allows the client to submit a UDDI inquiry and receive a UDDI response, while not concerning themselves with the details of the registry implementation. Firstly, UDDI inquiry requests are mapped to WRS Query requests. The UDDI entities [UDDI-DATA] referenced in the request are mapped to ebRIM entities [ebRIM], upon which the registry model is based. Secondly, the WRS response is mapped back to a valid UDDI message, with corresponding entity mappings.

This document exposes some of the details of these mappings. Not all aspects of the UDDI inquiries are supported due to inherent differences between UDDI and ebRIM. Enhanced functionality for mappings, including UDDI publisher capability, may be provided at a later time. For further information on any aspect of the mappings, the reader should refer to the references in Appendix A.

8.5.2 Entity Mapping

This section provides a simple overview of the mapping that is defined between UDDI entities and ebRIM entities. Any relevant notes or exceptions are also described. For more information, the reader should refer to the JAXR 1.0 Specification [JAXR], upon which the mappings are based.

UDDI Entity	ebRIM Entity	Notes
address	PostalAddress	
businessEntity	Organization	
businessInfo	Organization	A businessInfo instance contains only basic information on the business
businessService	Service	
bindingTemplate	ServiceBinding	
categoryBag	Collection of Classification instances	
categoryBag/keyedReference	Classification	
contact	User	

discoveryURL	ExternalLink	
identifierBag	Collection of ExternalIdentifier instances	
identifierBag/keyedReference	ExternalIdentifier	
overviewDoc	ExternalLink	
serviceInfo	Service	A serviceInfo instance contains only basic information on the service
tModel	ClassificationScheme	
tModelBag	Collection of ExtrinsicObject instances	Only applies to find_binding
tModelBag/tModelKey	ExtrinsicObject	Only applies to find_binding
tModelInfo	ClassificationScheme	A tModelInfo instance contains only basic information on the tModel
tModelInstanceInfo	SpecificationLink	Part of a bindingTemplate

8.5.3 Inquiry Mapping

This section provides more information on individual request mappings from a UDDI inquiry to WRS Query as well as information on the response transformation. Limitations, exceptions and other relevant pieces of information are discussed. Except where noted, all the UDDI inquiry functionality is supported by the registry. For detailed information on UDDI inquiry functions, the user should refer to the UDDI v.2 specification [UDDI-API].

General Notes

- There is currently no support for find_BusinessDetailExt.

- There is no support for certain <findQualifier> values: caseSensitiveMatch, sortByNameAsc, sortByNameDesc, sortByDateAsc, sortByDateDesc, orLikeKeys, combineCategoryBags and serviceSubset.

- The <findQualifier> values sortByDateAsc and sortByDateDesc could not be implemented in the stylesheet because it would require obtaining the audit trail for the relevant object.

- The maxRows attribute in the UDDI inquiry is not mapped to the WRS Query.

- The inquiry functions find_binding and get_bindingDetail return equivalent information. There is no bindingInfo element in UDDI, so the entire bindingTemplate is returned in both cases.

- The findQualifier exactNameMatch is mapped to ogc:PropertyIsEqualTo. Without this qualifier, the element ogc:PropertyIsLike is used with leading and trailing wildcards.

- The attribute keyName in categoryBag/keyedReference is not significant and is therefore ignored in requests.

8.5.3.1 Find Operations

find_binding

- There is no support for any <findQualifier> element.

- In the request, tModelBag/@tModelKey is mapped to the specificationObject attribute of the SpecificationLink.

- In the response, tModelInstanceInfo/@tModelKey points to an ExtrinsicObject such as a WSDL document that has the technical specifications for accessing the Service through the ServiceBinding.

- The property accessPoint/@URLType is found from the prefix of the accessURI in the ServiceBinding instance (e.g. **http** in http://www.galdosinc.com).

find_business

- There is no support for <tModelBag>.

8.5.4 find_relatedBusinesses

- There is no support for any <findQualifier> element.

- There is no support for <keyedReference>.

- Checks Associations for targetObject or sourceObject with id equal to the provided businessKey and returns the Organization at the other end of the Association.

- According to the UDDI v.2 spec, the association between the two businessEntity instances must be affirmed, i.e. confirmed by both the sourceOwner and targetOwner. Within the context of ebRIM, this requires that the attributes isConfirmedBySourceOwner and isConfirmedByTargetOwner of the Association are both set to true.

8.5.5 find_service

- There is no support for <tModelBag>.

find_tModel

- This operation returns only ClassificationScheme instances.

8.5.5.1 Get Operations

get_bindingDetail

- This query is fully supported.

get_businessDetail

- This query is fully supported.

- The property contact/@useType has no match in ebRIM and is therefore left blank.

- Only the “primaryContact” User is included, the Associations aren’t traversed for other Users.

- The property discoveryURL@useType is required but may contain an empty string if the ExternalLink instance has no Name.

get_serviceDetail

- This query is fully supported.

get_tModelDetail

- This operation returns only ClassificationScheme instances.

- According to the UDDI spec, there can be only one overviewDoc, so only the first ExternalLink is used in the response.

The operation get_tModelDetail returns overviewDoc (ExternalLink), identifierBag (ExternalIdentifier) and categoryBag (Classification) instances, even though it is unclear whether all these are relevant with respect to a ClassificationScheme in ebRIM.

References

An example list of references. Please update with appropriate and current references.

SA
Topic 12, Service Architecture. OGC Abstract Specification**,** available online: *http://www.opengis.org/public/abstract/01-112.pdf*

WMS
OpenGIS® Web Map Server Interfaces Implementation Specification, available online: *http://www.opengis.org/techno/specs/01-047r2.pdf*

GML
OpenGIS® Geography Markup Language (GML) Implementation Specification, available online: *http://www.opengis.net/gml/01-029/GML2.html*

SLD
OpenGIS® Styled Layer Descriptor (SLD), Discussion Paper, available online: *http://www.opengis.org/techno/discussions/01-028.pdf*

WTS
OpenGIS® Web Terrain Server (WTS), Draft Interoperability Program Report, available online: *http://ip.opengis.org/mpp1/docs/WTS_v0.3.1.zip*

XIMA
OpenGIS® XML for Imagery and Map Annotations (XIMA), Discussion Paper, available online: *http://www.opengis.org/techno/discussions/01-019.pdf*

Annexes

Annex A: Use Cases

Use detail for the user scenarios described in the text

Use Case 1

Use Case Description	
Name	Discover OGC Registries Through UDDI
Priority	High
Description	This use case concentrates on discovery of specialized OGC registries from UDDI registries, rather than direct discovery of OGC services themselves.
Precondition	Assumes that one or more UDDI registries and clients exist, and that one or more OGC registries and associated businesses have been published to them.
Flow of Events – Basic Path	
1.	User searches UDDI for businesses with OGC registry services
2.	User searches for OGC catalog/registries associated with businesses matching search criteria
3.	User makes use of WSDL associated to selected catalog/registry service to select/configure an appropriate OGC registry client.
4.	User switches to an OGC registry client to search for OGC services and/or geospatial content of interest.
5.	See Common Architecture / OGC registry use cases for continuations of this user scenario.
Flow of Events – Alternative Paths	
Step 3	User reads selected registry WSDL into a rapid development tool to create an OGC registry client or client components.

Postcondition	OGC registries published to UDDI contain up-to-date information on OGC services and content of interest to the user.

Use Case 2

Use Case Description	
Name	Discover OGC Services Through UDDI
Priority	High
Description	This use case concentrates on discovery of OGC services (including registries) and/or the businesses which provide them, through UDDI interfaces
Precondition	Assumes that one or more UDDI registries and clients exist, and that one or more OGC services and associated businesses have been published to them. Further assumes that sufficient service description / classification information is available within the UDDI registry to usefully narrow the list of services of interest.
Flow of Events – Basic Path	
1.	User searches UDDI for OGC service offerings with service types (tModels) of interest
2.	User narrows search using one or more geographic and non-geographic service and business classifications defined in the UDDI registry.
3.	User selects one or more OGC services of interest, obtains WSDL documents for binding information.
4.	User switches to an OGC client in order to discover the capabilities of services of interest.
5.	See Common Architecture / OGC registry use cases for continuations of this user scenario.
Flow of Events – Alternative Paths	
Step 1	User searches first for businesses and/or services using one or more geographic and non-geographic service and business classifications defined in the UDDI registry.

Step 2.	User reads selected service WSDL into a rapid development tool to create an OGC service client or client components.
Step 3.	Re-join at Step 4
Postcondition	Selected services offer content of interest to the user

Use Case 3

Use Case Description	
Name	Discover OGC Services Through UDDI interface on OGC registry
Priority	High
Description	This use case concentrates on discovery of OGC services (including registries) and/or the businesses which provide them, through an UDDI interface on an existing OGC registry
Precondition	Assumes that one or more OGC registries expose UDDI interfaces and that OGC services of interest are published to those registries.
Flow of Events – Basic Path	
1.	User searches UDDI for OGC service offerings with service types (tModels) of interest.
2.	User narrows search using one or more geographic and non-geographic service and business classifications defined in the UDDI registry or externally validated (perhaps from additional OGC registry information).
3.	User selects one or more OGC services of interest, obtains WSDL documents for binding information.
4.	User switches to an OGC client in order to discover the capabilities/content of the selected services of interest.
5.	See Common Architecture / OGC registry use cases for continuations of this user scenario.
Flow of Events – Alternative Paths	
Step 3	User searches UDDI registry for "human-facing" services (clients) of compatible types to services of interest.
Step 4.	User binds to the endpoints of services of interest with compatible clients to obtain service capabilities.

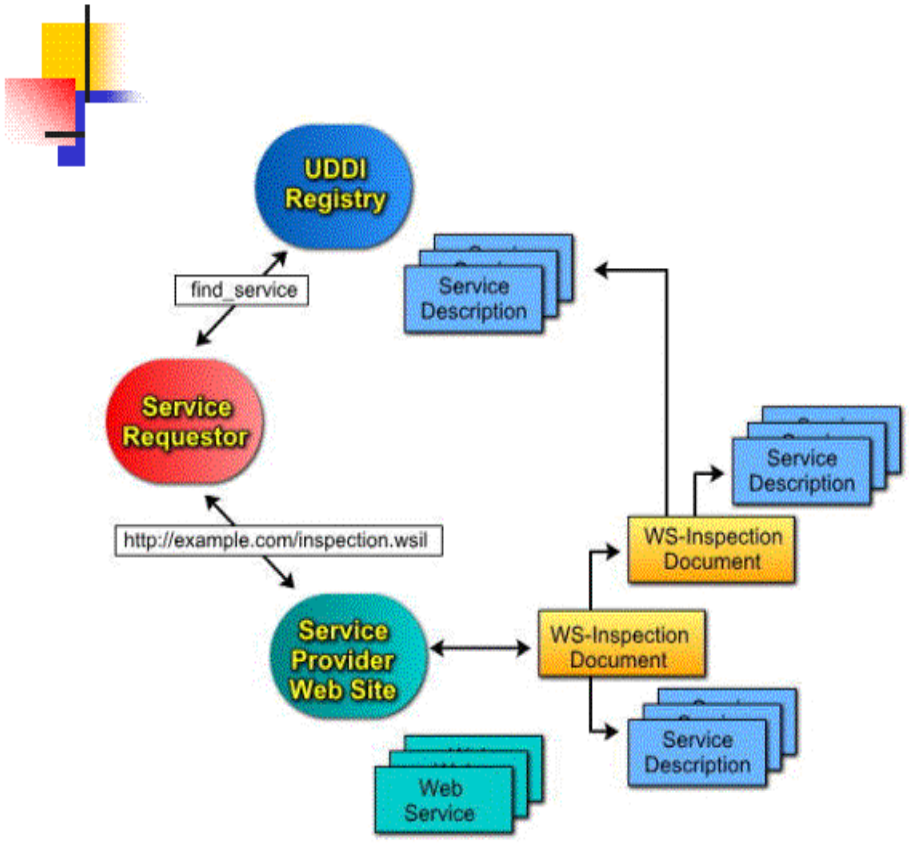
Postcondition	Selected services have content of interest which is usable through the selected clients.

Use Case 4

Use Case Description	
Name	Publish OGC Services To UDDI registry
Priority	High
Description	This use case concentrates on publishing of OGC services to a UDDI registry
Precondition	Assumes there are one or more UDDI registries and clients capable of publishing services, as well as OGC services available to be published.
Flow of Events – Basic Path	
1.	Publisher prepares needed information for registration of service into UDDI registry
2.	Publisher publishes business description to UDDI registry with UDDI client
3.	Publisher publishes service description to UDDI registry with UDDI client
4.	Publisher updates service description in UDDI registry with internally and externally validated classifications and categories.
Flow of Events – Alternative Paths	
Step 1.	Publisher publishes OGC service to OGC registry
Step 2.	OGC registry publishes service description automatically to UDDI registry, which may be integrated into the same registry service.
Step 3.	Rejoin at Step 4
Step 5.	UDDI registry publication triggers a service registration in a corresponding OGC registry
Postcondition	None

Use Case 5

This use case(s) will present event flows involving the use of Web Service Inspection Language in conjunction with UDDI to discover OCG Registries and Web Services



Annex B: UDDI V3/ebXML RegRep V2.3 Analysis

2.3Registry Functionality Comparison

Legend

Black Text defines in place functionality or specification.

Blue Text defines new functionality requirements

Red Text defines missing functionality or lack of concept of this functionality

Teal Text defines recognition of need but no plans to work.

Purple Text defines work in progress.

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
Search	**Search**	**Search**
Customer Browser Registry Interface	No direct interface to the registry defined.	No direct interface to the registry defined. APIs defined.
Application Interface –Search	Defined in ebXML Reg/Rep V2 8.0 Query Management Services (Query Management Client)	DEFINED in UDDI V3 4.8 Success and Error Reporting 5.1 Inquiry API Set 9 Policy 10 Multi-Version Support
Search information asset Records - Simple	DEFINED in ebXML Reg/Rep V2 8.1Ad-hoc Query	No concept of Information Asset in UDDI V3. See Search Web Service Records

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
Search Information Asset Record – Advanced	DEFINED in ebXML Reg/Rep V2 8.1 Ad-hoc Query can be restricted by return type values	No concept of Information Asset in UDDI V3. See Search Web Service Records
Search - Domain records	ebXML Reg/Rep V2 can find records of any type via Ad hoc queries 8.0 Query Management Services No concept for Domains as a partition of ownership of registry objects in ebXML Reg/Rep V2. 5.3 Registry Users	Service records categorized by businessEntity which is domain-like concept. DEFINED in UDDI V3 5.1 Inquiry API Set

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
NF - Search - Namespace records	ebXML Reg/Rep V2 can find records of any type via Ad hoc queries 8.0 Query Management Services No concept/support for Namespace registration and management	No support for Namespace registration and management Uses Namespaces to support API behavior and as attribute to categorize UDDI registry entities.

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
NF - Search - Web Services records	ebXML Reg/Rep V2 can find records of any type via Ad hoc queries 8.0 Query Management Services ebXML RIM V2 provides structure see 6.5 Service, 6.6 Service Binding, & 6.7 Service Link. Services can be registered as an object in ebXML Reg/Rep V2 in 7.3 Submit Objects Protocol.	DEFINED in UDDI V3 5.1 Inquiry API Set

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
NF - Search – Dependency Records	ebXML Reg/Rep V2 can find records of any type via Ad hoc queries 8.0 Query Management Services No concept/support for Dependency registration and management	No support for Dependency registration and management
NF - Search – Event Records	ebXML Reg/Rep V2 can find records of any type via Ad hoc queries 8.0 Query Management Services No concept/support for Event registration and management	No support for Event registration and management
Display/Retrieve	**Display/Retrieve**	**Display/Retrieve**

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
Customer Browser Registry Interface	No direct interface to the registry defined.	No direct interface to the registry defined. APIs only.
Browser Interface to Access/Retrieval 3A Service – Login	No direct interface to the registry defined.	No direct interface to the registry defined. APIs defined.
Application Interface to Access/Retrieval 3A Service – Login	DEFINED in ebXML Reg/Rep V2 9.7 Access Control	Defined in UDDI V3 5.3 Security Policy 9 Policy
Application Interface – Display/Retrieve	DEFINED in ebXML Reg/Rep V2 8.4 Content Retrieval	DEFINED in UDDI V3 Retrieval is metadata record information and not objects in a repository 9 Policy 10 Multi-Version Support

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
Sort Search Display	**Client side functionality – out of scope of spec.**	**Sort provided for UDDI registry information model items** **5.1 Inquiry API Set**
Select Record Detail Display	**Client side functionality – out of scope of spec.**	**Retrieval of metadata for consumption. Actual display is client side functionality.** **5.1 Inquiry API Set**

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
Retrieve Object via URL	DEFINED in ebXML Reg/Rep V2 8.4 Content Retrieval	The WSDL Technical Note wsdl-TN-V2.00-Draft-20020926[**Error! Bookmark not defined.**] provides a means to structure the Binding registry schema to accommodate URI links to resources and dccumentation.
Subscribe	**Subscribe**	**Subscribe**
Browser Interface – Subscription Login	No direct interface to the registry defined.	No direct interface to the registry defined.. All Interfaces to Registry are via API.

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
Application Interface – Subscription Login	IN WORK FOR ebXML Reg/Rep V3	DEFINED IN UDDI V3 4.8 Success and Error Reporting Subscription API Set 9 Policy 10 Multi-Version Support
Subscribe - Information Asset Record	IN WORK FOR ebXML Reg/Rep V3	No concept for this in the UDDI registry.

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
NF - Subscribe - Namespace Record	**No concept/support for Namespace registration and management** **No Concept for registration of record (a name) which does not point to an object in repository. 7.0 Life Cycle Mgmt. Has relationship to another record in registry.**	**No support for Namespace registration and management**
NF - Subscribe - Web Service Record	IN WORK FOR ebXML Reg/Rep V3	DEFINED IN UDDI V3 5.5 Subscription API Set

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
NF - Subscribe – Event Record	No support for non-registry (interfaced) Event registration and management No Concept for registration of record (a event) which does not point to an object in repository. 7.0 Life Cycle Mgmt. Has relationship to another record in registry. ebXML Reg/Rep V3 will allow subscription to any type of registry event on any registry object .	No support for Event registration and management

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
NF - Subscribe – Dependency • Events • Web Services • Namespaces • Information Assets	No support for Dependency registration and management No Concept for registration of record (a dependency) which does not point to an object in repository. 7.0 Life Cycle Mgmt. Has relationship to another record in registry.	No support for Dependency registration and management
Review Subscriptions -Information Asset Record	**IN WORK FOR ebXML Reg/Rep V3**	**No concept for this in the UDDI registry. See Review Subscription Web Services.**

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
NF - Review Subscriptions Namespace Record	**No support for Namespace registration and management**	**No support for Namespace registration and management**
NF - Review Subscriptions -Web Service Record	**IN WORK FOR ebXML Reg/Rep V3**	**DEFINED in UDDI V3** **5.5.7 Subscription API functions**
NF - Review Subscriptions – Event Records	**No support for Event registration and management**	**No support for Event registration and management**

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
NF - Review Subscriptions – Dependency Record • Events • Web Services • Namespaces • Information Assets	**No support for Dependency registration and management**	**No support for Dependency registration and management**
Un-Subscribe – Information Asset Record	**IN WORK FOR ebXML Reg/Rep V3**	**No concept for Information Asset in UDDI Registry. See Unsubscribe Web Services.**
NF - Un-Subscribe – Namespace Record	**IN WORK FOR ebXML Reg/Rep V3**	**No support for Namespace registration and management**

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
NF - Un-Subscribe – Web Service Record	**IN WORK FOR ebXML Reg/Rep V3**	**DEFINED in UDDI V3** **5.5.7 Subscription API functions**
NF - Unsubscribe – Event Record	**No support for Event registration and management**	**No support for Event registration and management**
NF - Un-Subscribe – Dependency Record • Events • Web Services • Namespaces • Information Assets	**No support for Dependency registration and management**	**No support for Dependency registration and management**

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
Subscription Notification Processing	IN WORK FOR ebXML Reg/Rep V3	**DEFINED in UDDI V3** 5.5.7 Subscription API functions
Error Handling	**Error Handling**	**Error Handling**
Customer Reports Error in Registry Content	No concept for customer direct interface to the registry defined.	No concept for customer direct interface to the registry defined

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
Auto Logging of Registry Event Errors	DEFINED IN ebXML Reg/Rep V2 7.3.5 Error Handling 7.4.3 Error Handling 7.7.3 Error Handling 7.8.3 Error Handling 7.9.3 Error Handling	DEFINED IN UDDI V3 4.8 Success and Error Reporting 7.6 Error Detection and Processing 12. Error Codes
Record Metrics	**Record Metrics**	**Record Metrics**
Customer Interface - Log Registry Events	No concept for customer direct interface to the registry defined	No concept for customer direct interface to the registry defined

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
Application Interface – Log Domain Registry Events	IN WORK ebXML Reg/Rep V3 Called Cooperating Registries	DEFINED IN UDDI V3 4.8 Success and Error Reporting 7.2.3 Change Record Journal 7.2.5 Replication Messages
Application Interface - Log Registry Events	DEFINED IN ebXML Reg/Rep V2 7.3.3 Audit Trail 7.4.1 Audit Trail 7.7.1 Audit Trail 7.8.1 Audit Trail	DEFINED IN UDDI V3 4.8 Success and Error Reporting 7.2.3 Change Record Journal 7.25 Replication Messages 7.6 Error Detection and Processing 12 Error Codes

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
Registration Audit Trail	DEFINED IN ebXML Reg/Rep V2 7.3.3 Audit Trail 7.4.1 Audit Trail 7.7.1 Audit Trail 7.8.1 Audit Trail	DEFINED IN UDDI V3 4.8 Success and Error Reporting 7.6 Error Detection and Processing 12 Error Codes
Register	**Register**	**Register**
Customer Browser Registry Administrative Login Interface	No direct interface to the registry defined.	No direct interface to the registry defined.

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
Domain Registry Application Interface – Administrative Login	IN WORK ebXML Reg/Rep V3 Called Cooperating Registries	DEFINED IN UDDI V3 7.4 Replication API Set 7.7 Validation of Replicated Data 8.0 Publishing Across Multiple Registries 9.0 Policy 10 Multi-Version Support

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
Creating new Information Record	DEFINED IN ebXML Reg/Rep V2 7.0 Life Cycle Management Service • Submit Objects • Update Objects • Add Slots • Association to Submitting Organization • Has no object versioning	No Concept for Information Asset Record IN UDDI V3. See Create Web Service Record.

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
Taxonomy Categorization Scheme for records, objects & elements - Add Taxonomy Terms	Has a placeholder to add common categorization scheme metadata to records & objects. Allows for multiple categorization schemes to be used. ebXML RIM V2 6.6 Classification Scheme 10 Classification of Registry Object. But does not identify the unifying scheme to use across all ebXML registries to support federated searches. No current scheme in place to categorize down to the element level.	Has a placeholder to add common categorization scheme metadata to registry entities. Uses tModels to build associations between entities in registry. Uses Publisher Assertion to build associations between entities. But does not identify the unifying scheme to use across all UDDI registries to support federated searches. No current scheme in place to categorize down to the element level.

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
Thesaurus Categorization Scheme for records, objects & elements - Add Thesaurus Terms	Has a placeholder to add thesaurus categorization terms metadata to records & objects Allows for multiple categorization schemes to be used. ebXML RIM V2 6.6 Classification Scheme 10 Classification of Registry Object. But does not identify the unifying scheme to use across all ebXML registries to support federated searches. No current scheme in place to categorize down to the element level.	Has a placeholder to add common categorization scheme metadata to registry entities. Uses tModels to build associations between entities in registry. Uses Publisher Assertion to build associations between entities. But does not identify the unifying scheme to use across all UDDI registries to support federated searches. No current scheme in place to categorize down to the element level.

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
Editing unfinished Records	No concept/support for Pending Record IN ebXML Reg/Rep V2	No Concept for Pending Record IN UDDI V3
Approve Information Asset Record	No concept/support for Pending Record in ebXML Reg/Rep V2	No Concept for Pending Record in UDDI V3
NF - Register – Create a new Namespace Record	No support for Namespace registration and management	No support for Namespace registration and management
NF - Register - Create new Web service Record	Defined in ebXML Reg/Reg V2	DEFINED IN UDDI V3 7.2 Concepts and Definitions DEFINED IN UDDI V3 7.5 Replication Configuration
NF - Register – Create new Event Record	No support for interface Event registration and management	No support for Event registration and management

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
NF - Approve Web Services Record	No Concept for Pending Record in ebXML Reg/Rep V2	No Concept for Pending Record.
NF - Approve Namespace Record	No Concept for Pending Record in ebXML Reg/Rep V2	No Concept for Pending Record in UDDI V3
NF - Approve Event Record	No support for Event registration and management	No support for Event registration and management
Register Information Asset Record	DEFINED IN ebXML Reg/Rep V2 7.0 Life Cycle Management Service Approve Objects	DEFINED IN UDDI V3 7.3 Change Record Structures
NF - Register namespace Record	No support for Namespace registration and management	No support for Namespace registration and management

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
NF - Register Web Service Record	Defined in ebXML Reg/Rep V2 in 7.3 Submit Objects Protocol. Defined in ebXML RIM V2 6.15 Service, 6.16 Service Binding, 6.17 Service Link.	DEFINED IN UDDI V3 6.0 Node Operation 7.2 Concepts and Definitions
NF - Register Event Record	Defined in ebXML Reg/Rep V2	No support for Event registration and management

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
Register Domain Record	DEFINED IN ebXML Reg/Rep V2 5.3 Registry Users Submitting Organization establishes contract with Registration Authority outside of scope of specification. No specific concept of partitions which are named, like domains, to show ownership of registry objects in ebXML Reg/Rep V2. 5.3 Registry Users	DEFINED IN UDDI V3 5.4 Custody and Ownership Transfer API Set 6.0 Node Operation 7.8 Adding a Node to a Registry Using Replication

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
Taxonomy Categorization Scheme for partitioning Domains	No current scheme in place to categorize domains. Categorization of partitions occurs outside of scope of the registry. Processes to preclude overlapping partitions occurs outside of scope of registry. Has placeholder in the ebXML RIM V2 to hold such information in Registry User, Registry Client , Responsible Organization, & Submitting Organization	No current scheme in place to categorize down to the element level. Categorization of partitions occurs outside of scope of the registry. Processes to preclude overlapping partitions occurs outside of scope of registry. Has placeholder in the UDDI V3 specification RIM to hold such information in Publisher, DomainKey

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
NF - Register Domain Type (service, data, stds body)	No concept for typing domains in ebXML Reg/Rep V2	No concept for typing domains in UDDI V3. Only has one domain type of Service but is an assumed characteristic.
Register Domain Steward – Becoming a Domain Steward	No concept for Domain Steward in ebXML Reg/Rep V2. See Register Content Owner.	DEFINED IN UDDI V3 5.4 Custody and Ownership Transfer API Set
Register Content Owner – Becoming a Content Owner	DEFINED IN ebXML Reg/Rep V2 5.3 Registry Users Responsible Organization has permission to create registry objects for Submitting Organization	No concept for Content Owner in UDDI V3. See Register Domain Steward.

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
Administrative Report Error	**Administrative Report Error**	**Administrative Report Error**
Report Error	Client functionality. Outside of scope for this specification.	DEFINED IN UDDI V3 7.6 Error Detection and Processing 12 Error Codes
Administrative Error Handling	**Administrative Error Handling**	**Administrative Error Handling**
Error Handling Edit/Delete Domain Record	Registration Authority is client side functionality outside of the scope of the specification.	DEFINED IN UDDI V3 4.8 Success and Error Reporting

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
Error Handling Edit/Delete Information Asset Record	DEFINED IN ebXML Reg/Rep V2 7.0 Life Cycle Management Service • Deprecate Objects • Remove Objects	No concept for Information Asset in UDDI V3
NF - Error Handling Edit/Delete Web Service Record	Defined in ebXML Reg/Rep V2	DEFINED IN UDDI V3 4.8 Success and Error Reporting 7.6 Error Detection and Processing 12 Error Codes
NF - Error Handling Edit/Delete Namespace Record	No support for Namespace registration and management	No support for Namespace registration and management

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
Error Handling Edit/Delete Event Record	No support for Event registration and management	No support for Event registration and management
Administrative Reporting	**Administrative Reporting**	**Administrative Reporting**
Reporting on Unapproved records report	No concept for Pending Record in specification in ebXML V2.	No concept for Pending Record in specification in UDDI V3
Reporting on Unfinished records	No concept for Pending Record in specification in ebXML V2	No concept for Pending Record in specification in UDDI V3
Reporting on Unregistered records report	No concept for Pending Record in specification in ebXML V2	No concept for Pending Record in specification in UDDI V3
Administrative Profiles	**Administrative Profiles**	**Administrative Profiles**

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
Profiles – Access - Managing Access Profiles (for CENTRAL roles only)	No direct interface to the registry defined. Responsibility of Responsible Organization. See 5.3 Registry Users. ebXML Reg/Rep V2 has Procedures defining access for roles defined. 9.7 Access Control See 2 below	No direct interface to the registry defined. DEFINED IN UDDI V3 Policies for overall administration specified in 5.4 Custody and Ownership Transfer API Set 6.0 Node Operation 7.8 Adding a Node to a Registry Using Replication 9.0 Policy
Administrative Search	**Administrative Search**	**Administrative Search**

Required **System Functionality** **to support Data Sharing** (**NF -** represents new registry functionality requirements)	**EbXML V2 Service and RIM Specification Functionality** http://www.oasis-open.org/committees/regrep/	**UDDI Specified Functionality** http://uddi.org/pubs/uddi-v3.00-published-20020719.pdf
Search System Record - system log	DEFINED IN ebXML Reg/Rep V2 8.2.5 Auditable Event Query	DEFINED IN UDDI V3 7.2.2 Change Record 7.2.3 Change Record Journal 7.2.4 High Water Mark Vector 7.3 Change Record Structures

CENTRAL Registry

Policies must be followed to establish a Domain. Domain and Domain Stewards must be registered in CENTRAL. Content Owners, Domain Stewards, and Registrars perform Life Cycle management on their assets, according to their roles.

Architecture Subcouncil and CENTRAL Change Board, according to their roles, set overall administrative policies for the Registry System (hub & domain) for handling records and objects. Domain Stewards and Content Owners are authorized to manage records and associated objects.

. WSSO handles authentication for CENTRAL. CENTRAL currently handles authorization for asset record creation and editing.

- **Customers (Users – people and applications) can look at any record and object which is not shielded by 3A policy established by domain. Customers need to get registered in 3A service to gain permissions to see shielded records and objects. This is not currently correct. We have implemented WSSO, which requires authentication before searching and viewing asset records. This is the first step in the functionality of shielding records.**

- **Profiles**

CENTRAL Org Registrar(s) – Can perform all Life Cycle functions for any domain.

Domain Steward & Content Owner – Can perform role-specific life cycle functions except altering registered records. Must be member of a Domain.

Subscriber – can subscribe to and unsubscribe to asset records

User – has no CENTRAL profile. Read only access to registry records and objects. Note that implementation of WSSO will require all Registry users to authenticate before accessing registry.

Oasis ebXML Registry V2 Specification

Registry Clients establishes a contract with the Registry Authority to submit records and objects. Role based permissions.

V2 spec set the overall administrative policy for registry system for handling records and objects.

Registry of the Registry User and Registry Guest outside of scope of V2 spec.

- **Registry Administrator sets security policy for records and objects in Registry Authority. Submitting Organization empowers Responsible Organization to do Life Cycle Management on records and objects; must be Registered Users to perform functions. Registry Guest can read records and may look at some objects. Registry Reader has read**

- **Profiles**

Registry Administrator – access to all methods on all registry (records &) objects.

Content Owner – same as Submitting Organization. Can perform LC Mgmt on records he owns. Must be a Registered User.

- **Registry Guest – has no profile. Read only access to registry records and objects.**

Categorization

Registry Information Model 6.6 Classification Scheme

ClassificationScheme instances are RegistryEntry instances that describe a structured way to classify or categorize RegistryObject instances. The structure of the classification scheme may be defined internal or external to the registry, resulting in a distinction between internal and external classification schemes.

UDDI V3 Specification

Though the UDDI Specification makes extensive use of Namespaces to support registry processing and application interface behaviors based on the Namespace identifier value. It does not register or manage namespaces.

Namespace: A collection of distinct names represented as strings of characters. Usually the names in a namespace are constructed according to a set of rules given by the definition of the namespace. URIs of various kinds are commonly used to construct the names in namespaces. For example, the namespace for UDDI keys in the recommended keying scheme consists of the URIs in the "uddi" scheme.

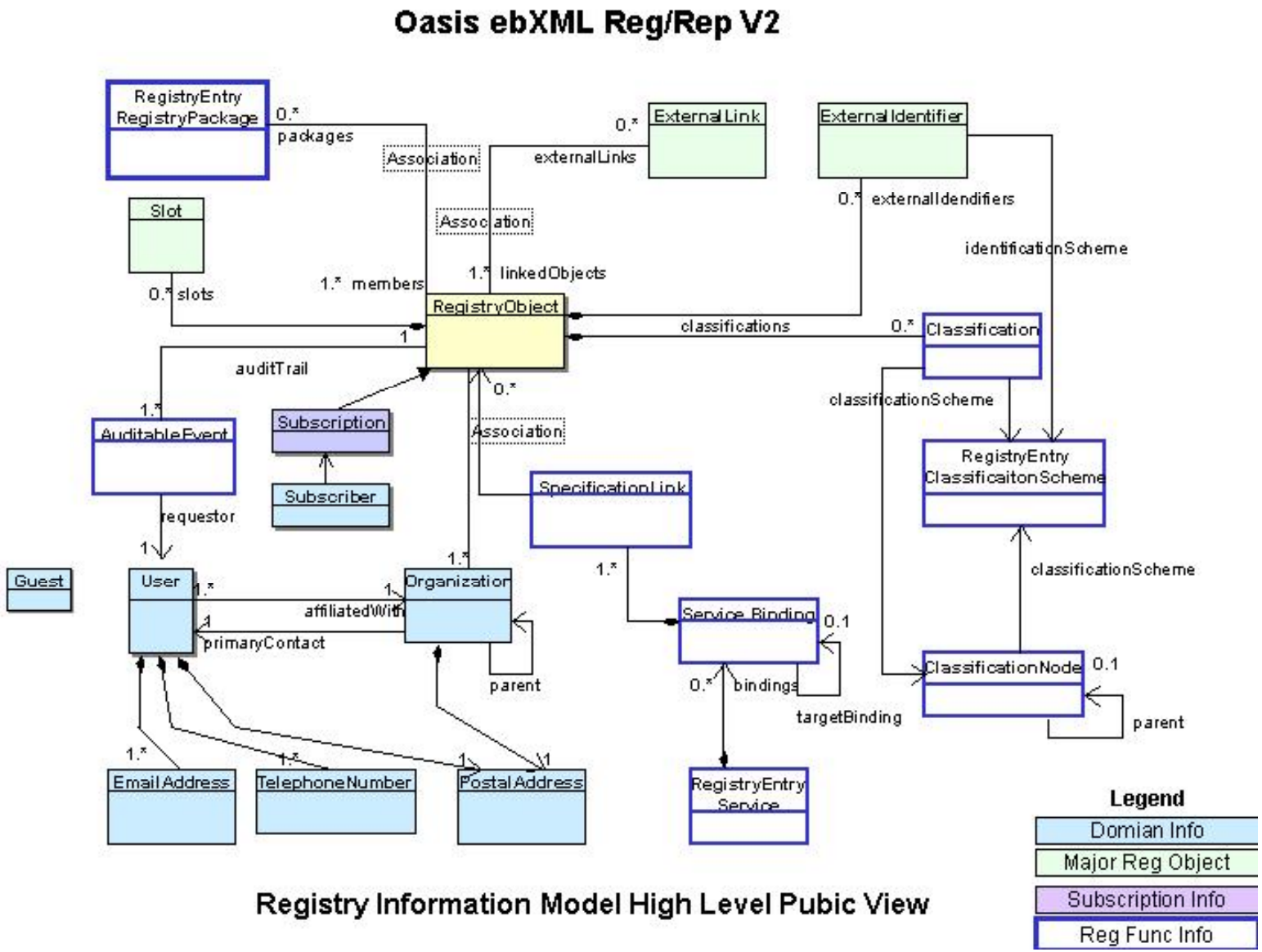
Namespaces are used to provide references to indicate UDDI version and the support or default behavior of the registry and the API sets. Namespace used as criteria to support search, identification, categorization, versioning, and behaviors. UDDI specifies a naming convention to use in construction of UDDI Namespace names used in Schema versioning. UDDI specifies the behaviors for default Namespace support.

Schema Versioning

XML prefix conventions – default namespace support

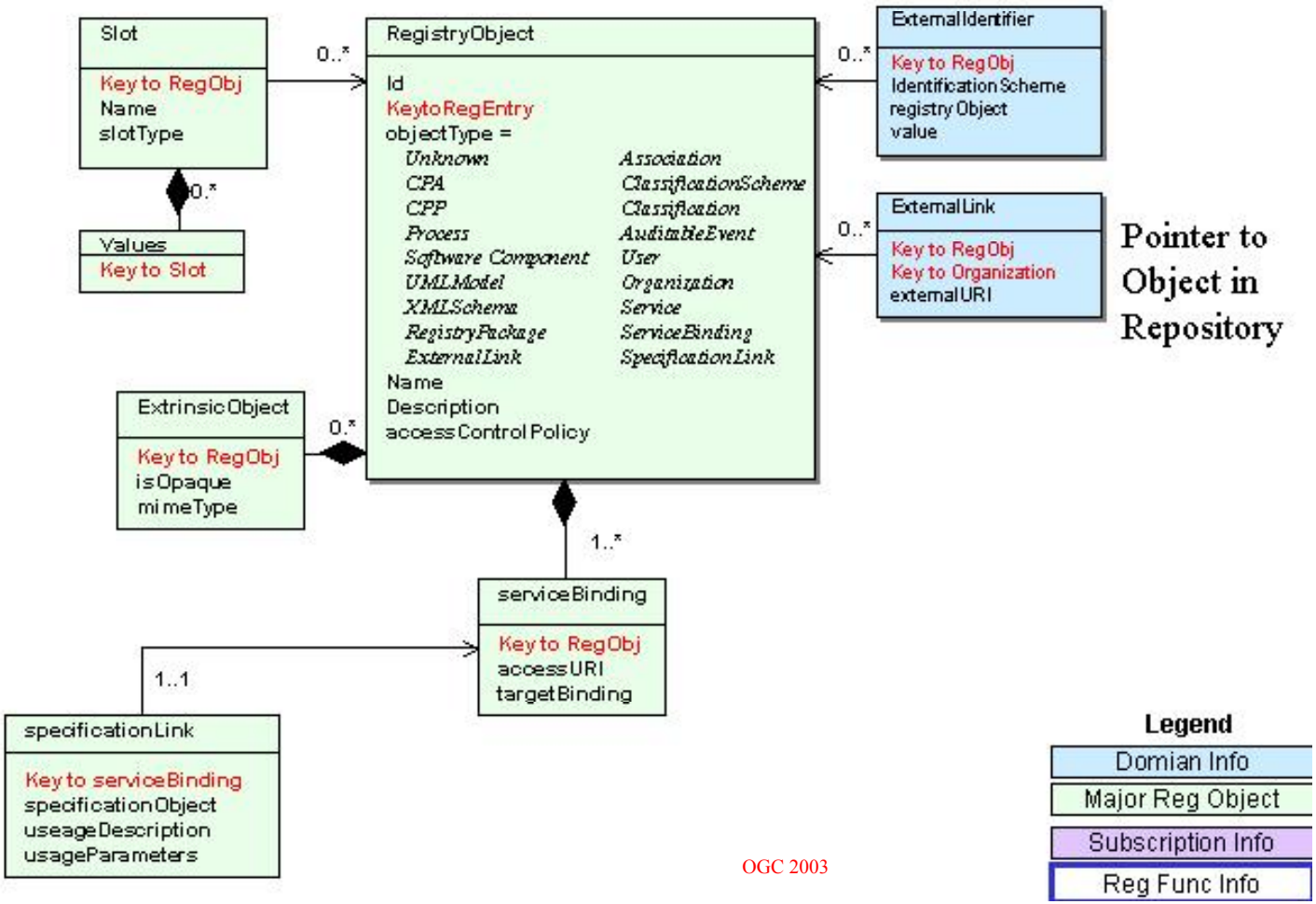
Registry Information Model Comparison

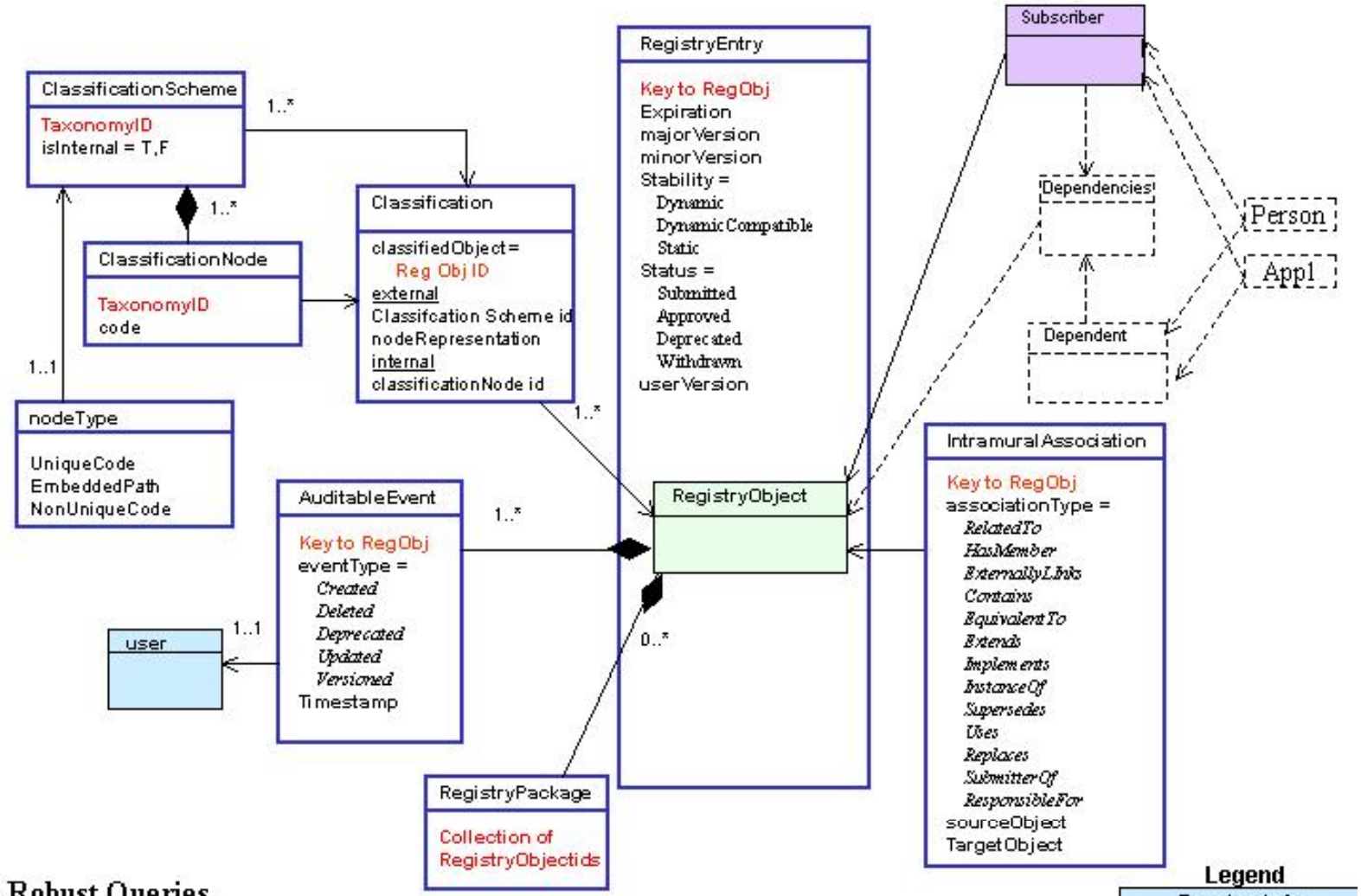
ebXML Registry Information Model (RIM)



Registry Information Model High Level Pubic View

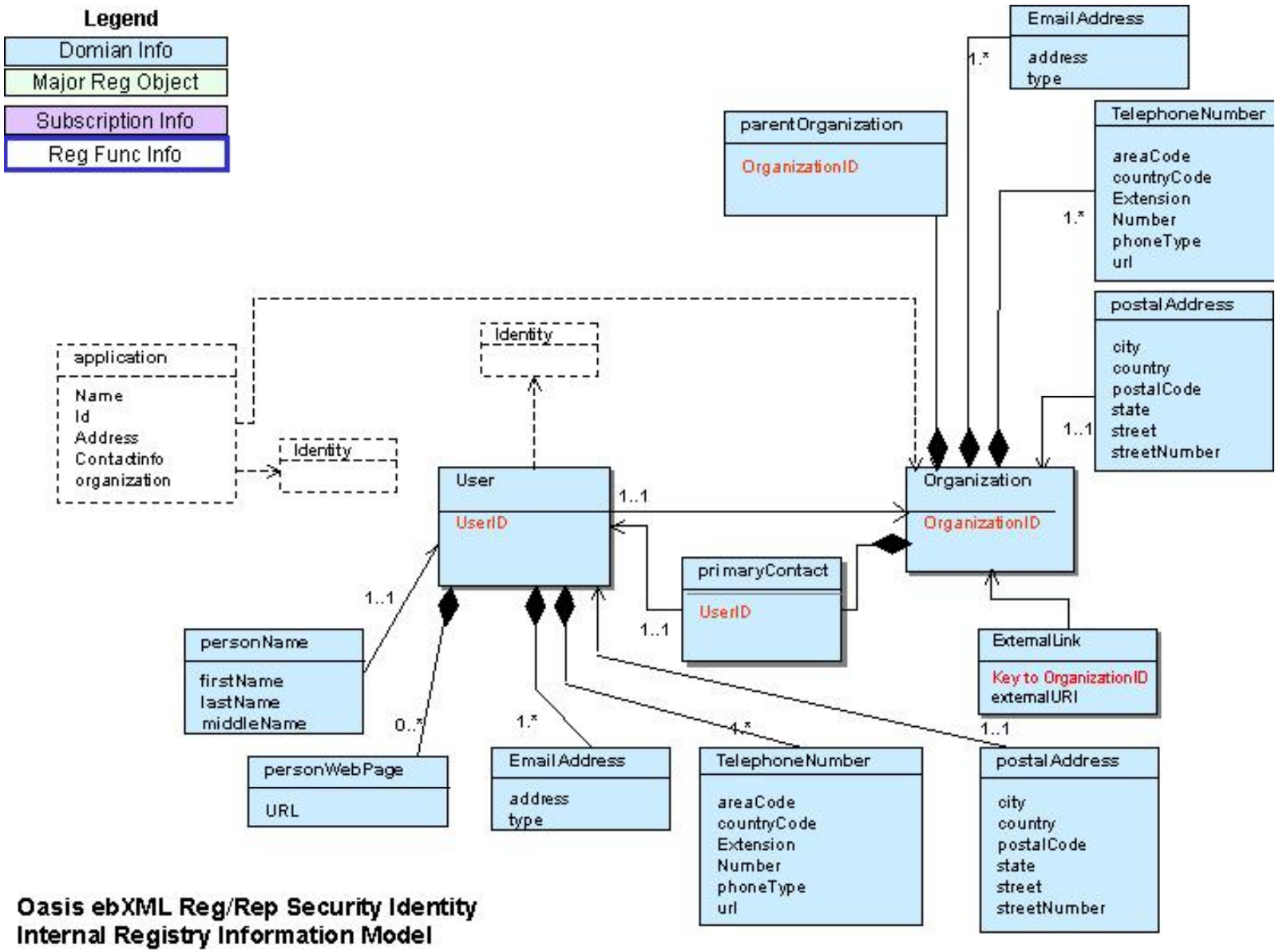
Oasis ebXML Reg/Rep RegistryObject Public Information Model

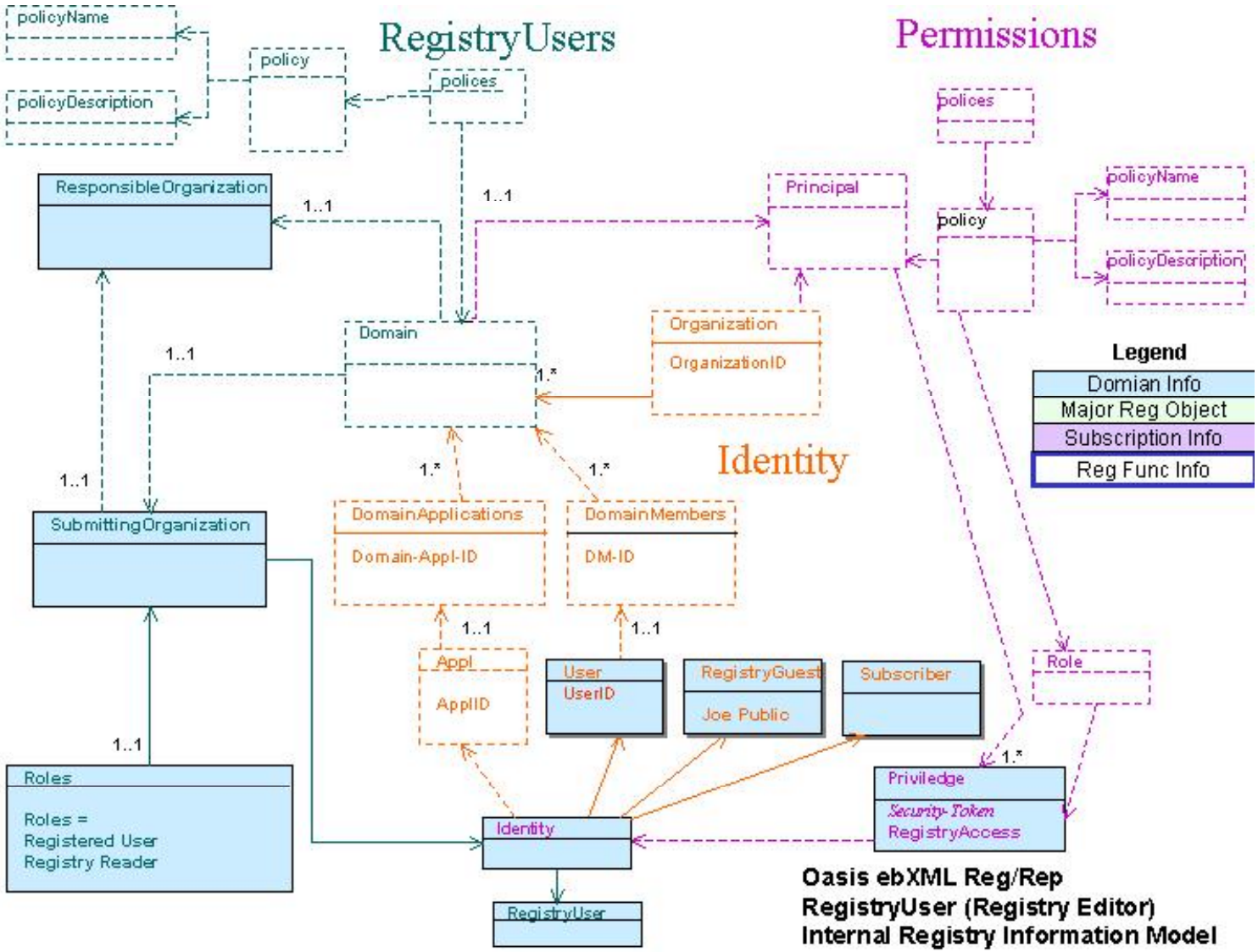


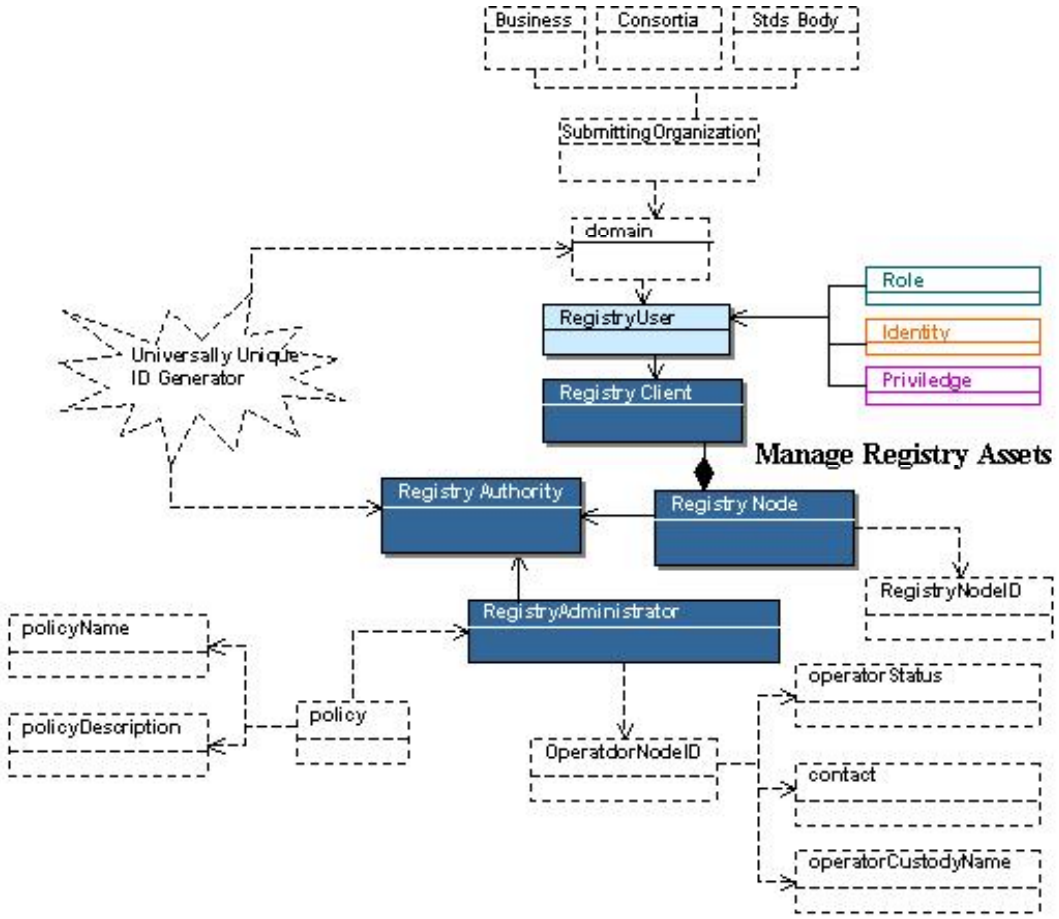


Robust Queries
Complex Content Retrieval

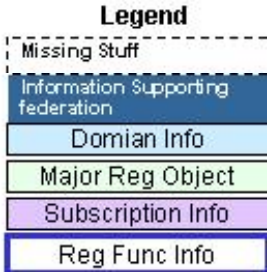
Oasis ebXML Reg/Rep RegistryObject
Internal Registry Information Model


Legend
Domian Info
Major Reg Object
Subscription Info
Reg Func Info
EmailAddress
address
type
1.*
parentOrganization
OrganizationID
TelephoneNumber
areaCode
countryCode
Extension
Number
phoneType
url
1.*
postalAddress
city
country
postalCode
state
street
streetNumber
1..1
Identity
application
Name
Id
Address
Contactinfo
organization
Identity
User
UserID
1..1
Organization
OrganizationID
primaryContact
UserID
1..1
1..1
personName
firstName
lastName
middleName
ExternalLink
Key to OrganizationID
externalURI
1..1
0..*
1.*
1.*
personWebPage
URL
EmailAddress
address
type
TelephoneNumber
areaCode
countryCode
Extension
Number
phoneType
url
postalAddress
city
country
postalCode
state
street
streetNumber
Oasis ebXML Reg/Rep Security Identity
Internal Registry Information Model

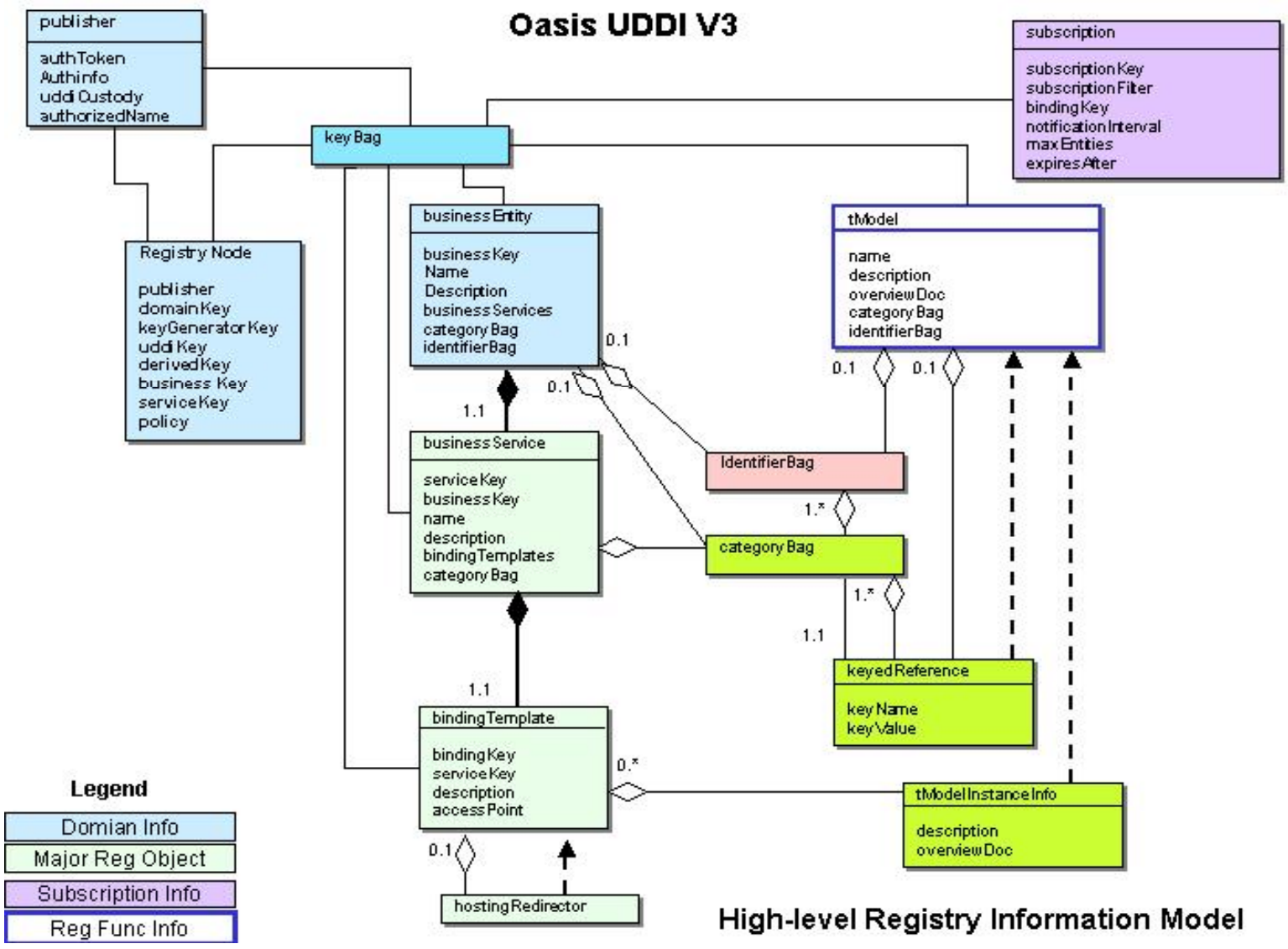

RegistryUsers
Permissions
Identity
policyName
policy
policyDescription
polices
ResponsibleOrganization
Principal
Domain
Organization
OrganizationID
SubmittingOrganization
DomainApplications
Domain-Appl-ID
DomainMembers
DM-ID
Appl
ApplID
User
UserID
RegistryGuest
Joe Public
Subscriber
Role
Roles
Roles =
Registered User
Registry Reader
Priviledge
Security-Token
RegistryAccess
Identity
RegistryUser
1..1
1..*
Legend
Domian Info
Major Reg Object
Subscription Info
Reg Func Info
Oasis ebXML Reg/Rep
RegistryUser (Registry Editor)
Internal Registry Information Model

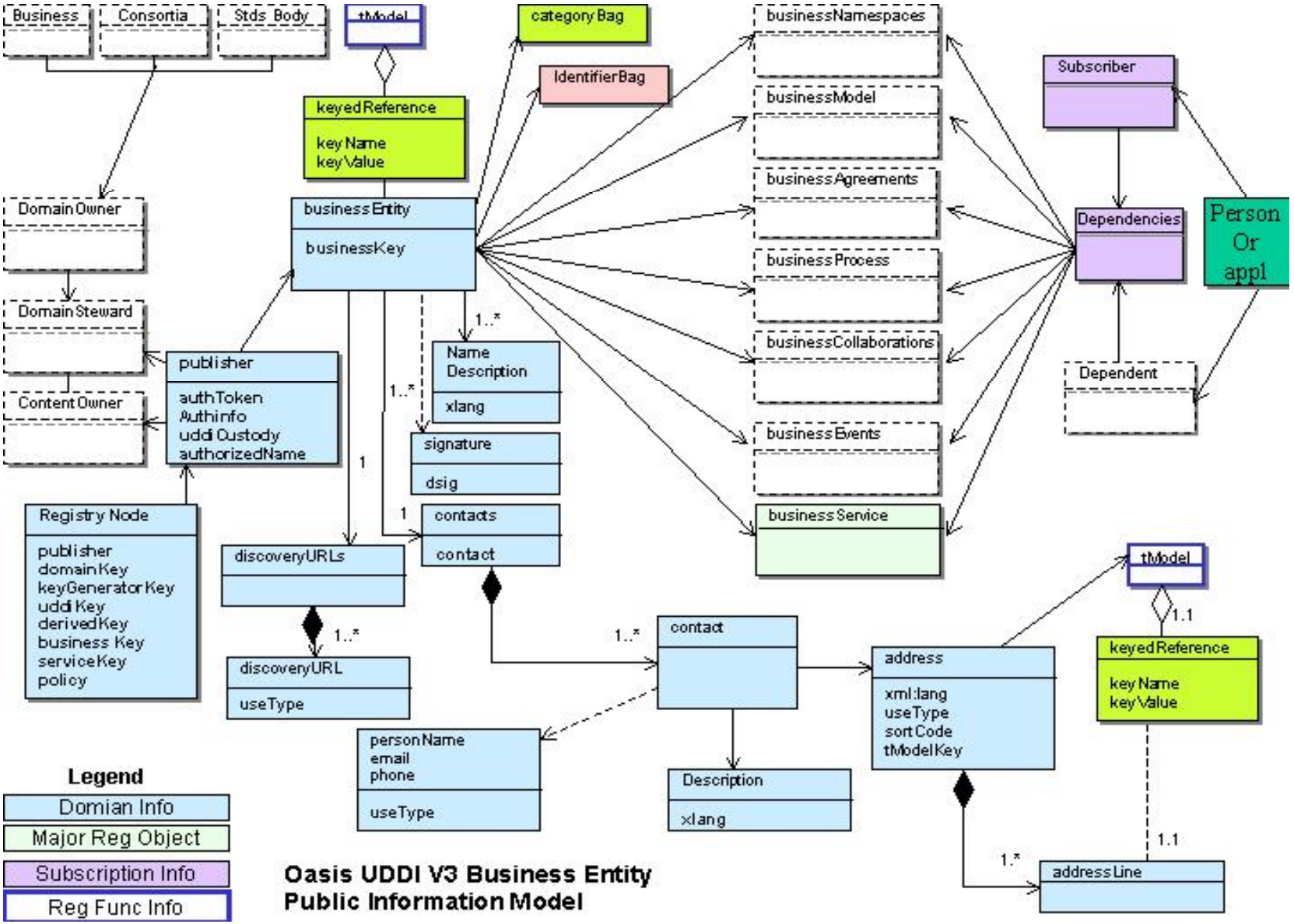


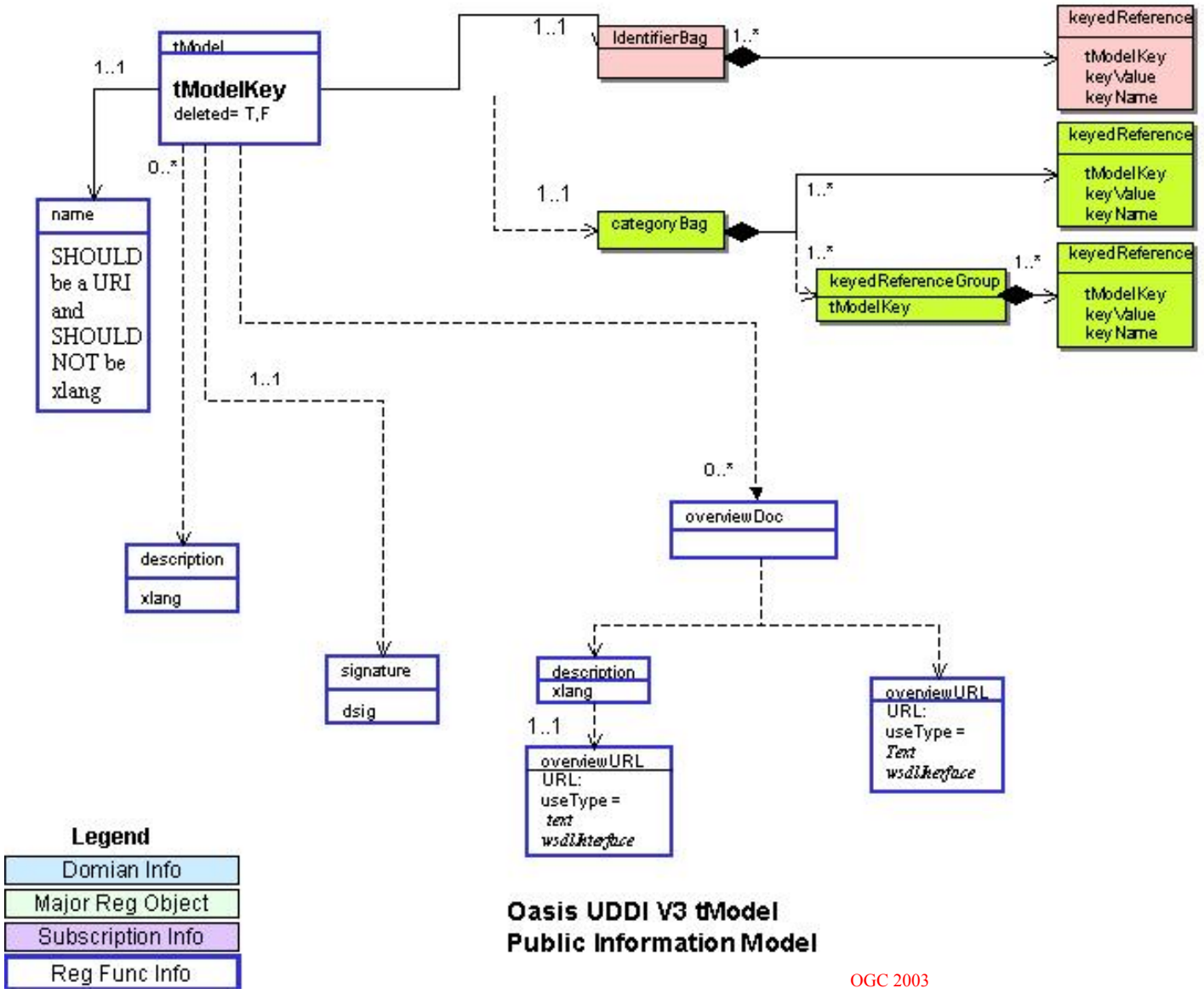
Oasis ebXML Ownership and Access
Internal Registry Information Model



UDDI Information Model


Oasis UDDI V3
publisher
authToken
Authinfo
uddi Custody
authorizedName
subscription
subscriptionKey
subscriptionFilter
bindingKey
notificationInterval
maxEntities
expiresAfter
keyBag
businessEntity
businessKey
Name
Description
businessServices
categoryBag
identifierBag
tModel
name
description
overviewDoc
categoryBag
identifierBag
Registry Node
publisher
domainKey
keyGeneratorKey
uddi Key
derivedKey
business Key
serviceKey
policy
0.1
0.1
0.1
0.1
1.1
businessService
serviceKey
businessKey
name
description
bindingTemplates
categoryBag
IdentifierBag
1.*
categoryBag
1.*
1.1
1.1
keyedReference
keyName
keyValue
bindingTemplate
bindingKey
serviceKey
description
accessPoint
0.*
tModelInstanceInfo
description
overviewDoc
0.1
hostingRedirector
Legend
Domian Info
Major Reg Object
Subscription Info
Reg Func Info
High-level Registry Information Model


Business
Consortia
Stds Body
tModel
categoryBag
IdentifierBag
businessNamespaces
businessModel
businessAgreements
businessProcess
businessCollaborations
businessEvents
businessService
Subscriber
Dependencies
Dependent
Person Or appl
keyedReference
keyName
keyValue
businessEntity
businessKey
DomainOwner
DomainSteward
ContentOwner
publisher
authToken
Authinfo
uddiCustody
authorizedName
Name
Description
xlang
signature
dsig
contacts
contact
discoveryURLs
discoveryURL
useType
RegistryNode
publisher
domainKey
keyGeneratorKey
uddiKey
derivedKey
businessKey
serviceKey
policy
personName
email
phone
useType
Description
xlang
address
xml:lang
useType
sortCode
tModelKey
addressLine
1..*
1
1.1
Legend
Domian Info
Major Reg Object
Subscription Info
Reg Func Info
Oasis UDDI V3 Business Entity
Public Information Model



Oasis UDDI V3 tModel
Public Information Model