Open Geospatial Consortium Inc.

Date: 02-MAR-2006

Reference number of this OGC® Project Document: **OGC 05-109r1**

Version: 0.0.4

Category: OpenGIS® Discussion Paper

Editor: Panagiotis (Peter) A. Vretanos (CubeWerx Inc.), Rento Primavera (Ionic)

Catalog 2.0 IPR for OWS 3

Copyright notice

Warning

This document is not an OGC Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard.

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® Discussion Paper
Document subtype:
Document stage: Approved
Document language: English

Contents

Tables and Figures

i. Preface

The Open Geospatial Consortium (OGC) is an international industry consortium of more than 300 companies, government agencies, and universities participating in a consensus process to develop publicly available geo-processing specifications. This Interoperability Program Report (IPR) is a product of the OGC Web Services, Phase-3 (OWS3) project.

The OGC Web Services Initiative, Phase 3, is part of the OGC's Interoperability Program: a global, collaborative, hands-on engineering and testing program designed to deliver prototype technologies and proven candidate specifications into the OGC's Specification Development Program. In OGC Interoperability Initiatives, international teams of technology providers work together to solve specific geo-processing interoperability problems posed by Initiative sponsors.

ii. Submitting organizations

This draft Interoperability Program Report – Engineering Specification is being submitted to the OGC Interoperability Program by the following organizations:

CubeWerx Inc.
200 rue Montcalm, Suite R-13
Gatineau, QC J8Y 3B5
Canada

Ionic Software
18, Rue de Wallonie
4460 Grace-Hollogne
Belgium

Document Contributor Contact Points

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

Panagiotis (Peter) A. Vretanos
CubeWerx Inc.

Renato Primavera
Ionic Software

Vincent Delfosse
Ionic Software

iii. Revision history

Date	Release	Description
31-MAY-2005	0.0.0	Initial version
10-AUG-2005	0.0.1	WMS Profile Revision
22-AUG-2005	0.0.2	Addition WFS and WCS
25-AUG-2005	0.0.3	Revision of WFS and WCS
03-MAR-2006	0.0.4	Addition of Roadmap, Common Profile, XML/UML schemas, Symbology Encoding. Re-sequence figures and tables.

iv. Changes to the OpenGIS® Abstract Specification

T.B.D.

v. Changed to the OpenGIS® Implementation Specifications

T.B.D.

Foreword

Attention is drawn to the possibility that some of the elements of this part of OGC 05-109 may be the subject of patent rights. Open Geospatial Consortium Inc. shall not be held responsible for identifying any or all such patent rights.

Introduction

The purpose of this document is to show how to map the various types of metadata documents to be used in the OWS3 project into the OASIS ebXML Registry Information Model – Version 3.0, hereafter known as ebRIM. The OASIS ebRIM specification document defines the types of metadata and content that can be stored in an ebXML Registry.

The ebRIM is also a catalogue information model that has been implemented by a number of vendors that are members of the OGC. This information model is operated upon using the API defined in the Catalogue 2.0.1 implementation specification. The types of metadata that are to be used in the OWS3 project include WMS capabilities documents, WFS capabilities documents, metadata documents describing GML application schemas and UML models, etc …

If the manner in which metadata is mapped into the registry information model is explicitly proscribed, then CSW 2.0.1 queries should be portable across catalogue implementations and a greater degree of interoperability between and catalogue clients and catalogue server implementations may be achieved. In addition, this document will describe how a catalogue should behave in certain situations is order to ensure that operations achieve the same result on all catalogues. A typical example of this type of guidance would be to describe a “cascaded delete” operation. Such a description might address the issue of what should occur when one of a set of related or associated items is deleted.

Catalogue 2.0.1 with Corrigendum IPR for OWS3

1 Scope

The purpose of this document is to show how to map information from metadata documents used to describe resources in OWS3 into the registry information model defined in the ebXML Registry Information Model (ebRIM) specification [1].

If all the relevant metadata is extracted from the source metadata documents and mapped into the ebRIM, a client application need only know how to query a single common model during discovery. This is in contrast to other possible approaches, which store the metadata as content in the registry's repository and query the metadata structure (possibly using different query methods such as xpath or SQL) directly to resolve queries.

In this document, the mapping between the capabilities documents and the ebRIM is mainly described using XPath expressions. Each path from the source document is mapped to an appropriate path in the XML encoded representation of the ebRIM.

In addition to XPath expressions, this document uses schema diagrams and narrative text to describe the mapping from the metadata documents to the ebRIM.

To understand the explanations in this document, it is useful to keep the classes that makeup the ebRIM in mind. Appendices A, B, and C present several key views of the registry that are reproduce here.

This document classified metadata documents into three broad categories:

1. Service metadata expressed as OGC capability documents

2. Data metadata expressed using well known, standard metadata formats such as FGDC, ISO19139, etc …

3. Other metadata that is already defined or will be defined during the OWS3 project (e.g. metadata for GML application schemas or UML models).

Clauses 6 and 7 discuss service metadata expressed as OGC capability documents. Service metadata expressed as OGC capabilities document merits special handling since such metadata documents typically describes not only the service being offered but also the content that the service operates on as well as additional information such as styling information or links to other standardized metadata. The general approach that this document takes for handling service capability documents is to, as far as is possible, map metadata information from source documents directly and completely into the ebRIM.

Clause 8 describes the mapping of the core queryable properties described in the CSW 2.0 specification [2] to ebRIM elements. Specifically the content of the element csw:Record is mapped into the ebRIM model.

Clause 9 discusses the ISO profile (OGC Document 04-038) and its relationship to the ebRIM profile (OGC Document 05-025). It is likely that a catalog that conforms to the ebRIM profile may also be able to conform to the ISO profile as well.

Clause 10 discusses usage notes that are relevant to processing source metadata document into an ebRIM registry. Such notes, for example, might describe how embedded metadata references should be handled by a registry by describing what ebRIM records should be created how those records should be associated and/or classified. Besides using narrative text to describe such usage notes, UML models may also be employed.

2 Conformance

Not required in an IP DIPR, IPR or Discussion Paper.

3 Normative references

The following normative documents contain provisions that, through reference in this text, constitute provisions of this Interoperability Program Report. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this document (OGC 03-041) are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies.

[1] Oasis Document, *Oasis/ebXML Registry Information Model, V2.5,* http://www.oasis-open.org/committees/regrep/documents/2.5/specs/ebrim-2.5.pdf, 2003

[2] OGC Document 04-021r2, *OpenGIS® Catalogue Services Implementation Specification, version 2.0.1*, http://portal.opengeospatial.org/files/?artifact_id=5929&version=1, 2004

[3] OGC Document 04-024, *OpenGIS® Web Map Service Implementation Specification, version 1.1.3*, http://portal.opengeospatial.org/files/?artifact_id=5316, 2004

[4] OGC Document 04-094, OpenGIS Web Feature Service Implementation Specification version 1.1.0, http://portal.opengeospatial.org/files/?artifact_id=8339, 2004

[5] OGC Document 03-065r6, OpenGIS Web Coverage Service Implementation Specification version 1.0.0, https://portal.opengeospatial.org/files/?artifact_id=3837

[6] Department of Defense Discovery Metadata Specification version 1.2

4 Terms and definitions

T.B.D.

5 Conventions

T.B.D.

5.1 Requirement levels

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119 [1].

6 Catalogue 2.0.1 specification context and roadmap

The OGC Catalogue 2.0.1 with corrigendum specification consists of a hierarchical family of specification documents ranging from normative references to OGC Abstract Specification topics and ISO 191* specifications, through conceptual interface models and platform protocols to various levels of resource metadata information model profiles.

6.1 Specification layers and documents

A basic picture of the catalogue specification structure is shown in figure XXX, from most general at the base to most specific to a domain and application at the top.

An important distinction should be drawn between information model profiles which are specific to a type of metadata resource (e.g. Web Map Service capabilities) and those which present a common picture of the relationships and commonalities between registered resources (e.g. classification of feature types, associations between datasets and services, title of a resource). The working terminology in this report for these types of information profiles is

- Resource metadata information profile (MIM) - the model of specific resource descriptions in a catalogue
- Registry information profile (RIM) - the model of how multiple and diverse resource descriptions are presented within a catalogue

Within each of these information profile types there may be additional hierarchy in which more general resource types (e.g. Data Schema) are profiled to define more specific resource types (e.g. DDMS).

6.2 Basic registry information model

What may be the most basic possible Registry Information Model for CS/W catalog implementations consists of one object, csw:Record and its Core Queryable attributes as introduced in [2]. This CS/W RIM and its relationship to ebRIM are discussed in Clause 11 of this report.

6.2.1 ebRIM registry architecture

6.2.1.1 Introduction

This clause discusses, in general terms, the components that make up an ebRIM-based, CS/W 2.0 compliant catalogue system. A block diagram is presented that contains all the components of a catalogue system: ebRIM model, CS/W server, content repository and perhaps content manager. Then, there is a discussion about how the components work together to implement a catalogue system.

6.2.2 Other registry information models

Other Registry Information Models have been defined elsewhere but not yet specified in a Catalogue 2.0.1 Profile. While the ISO Catalogue Profile [OGC Document 04-038r3] is not defined specifically as a RIM, it does describe a structure for relating services and content which might prove useful as a pattern of metadata record relationships intermediate in complexity between CS/W RIM and ebRIM.

6.2.3 Document roadmap for guidance in this report

The guidance and observations in the present report span a number of the described catalog specification layers. This content will be found most useful as a reference for interoperable implementation and operation of catalogues if it is incorporated into future documents and revisions as follows:

- Layered structure and information model types discussion -> Catalogue 2.0 CS/W Specification document
- Elaboration of CS/W RIM or Common Profile -> Catalogue 2.0 CS/W Specification document
- Refinement of ebRIM objects for Catalogue 2.0 implementations -> Catalogue 2.0 ebRIM Profile document
- Mapping of ebRIM onto CS/W RIM -> Catalogue 2.0 ebRIM Profile document
- Guidance on mapping of specific resources into ebRIM -> Domain-specific MIM Profile document(s), e.g.
    - OWS MIM

- Feature MIM
- SWE MIM
- Imagery MIM
- Etc.

It is specifically not recommended to include resource mappings directly into the ebRIM Profile except as examples. The motivation for this is to encourage greater stability in RIM Profile specifications, since these will have the greatest effect on the "implementability" and interoperability of catalogue clients and services.

7 ebRIM registry architecture

7.1 Catalogue components

Figure 1 illustrates the conceptual architecture of a catalogue server. There is a client or user agent of some type that sends requests to the catalogue that comply with the CSW 2.0 specification using HTTP as the distributed computing platform. A catalogue server that complies with the CSW 2.0 specification processes the request and sends a response back to the client, again using HTTP. In the course of processing the request, the catalogue interacts with its information model (IM), ebRIM is this case, and an optional repository. The information model and repository reside within some database system. Although figure 1 shows the IM and the repository existing in the same database, this does not necessarily have to be the case. No assumption is made about the nature of the database system. It can be an SQL-based relational database system, an object database or even an XML database.

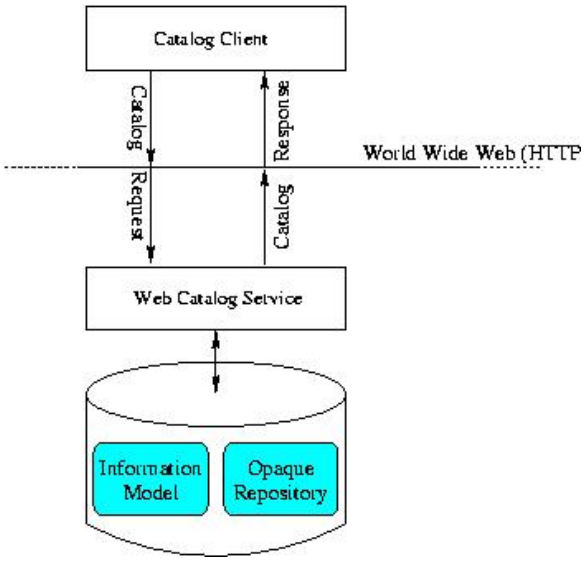


Figure 1 – Conceptual Architecture

7.2 Catalogue operation

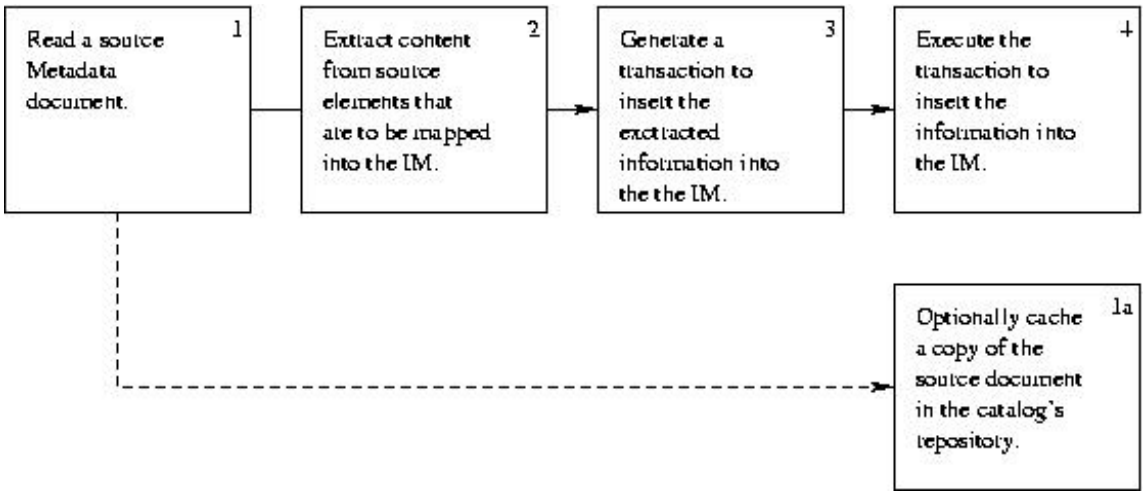
Catalogue requests are processed by the CSW 2.0 compliant catalogue server, which interacts with the ebRIM information model that is materialized in some database system. Information is inserted, update or deleted from the information model using the **Harvest**, and **Transaction** operations. Information can be retrieved from information model using the **GetRecords** or **GetRecordById** operation.

7.2.1 Inserting information into the catalogue

The **Harvest** or **Transaction** operations can be used to create information in the information model of the catalogue. The typical workflow for processing a metadata document by a catalogue service is presented in figure 2.

A metadata document is read and the content of each element to be inserted into the IM is extracted. A transaction is created to insert the extracted content into the IM. The transaction is executed to store the information in the database. Optionally, a copy of the source metadata document is cached in the catalogue's repository.

Figure 2 – Data creation workflow



If the metadata is being inserted into the catalogue using the **Harvest** operation, then steps 1 through 4 are automatically performed by the catalogue – assuming it knows the mapping from the source document to it IM. If it does not, then the catalogue shall throw an exception.

If the **Transaction** operation is being used to populate the catalogue's IM, then steps 1, 2 and 3 are performed by a user or a user agent and the catalogue only needs to process step 4. In this case, it is the user or user agent that needs to be aware of the mapping from the source metadata document to the catalogue's IM.

In either case, one of the main purposes of this document is to provide standard mappings from known source metadata documents into ebRIM, which is the IM of the catalogs used in the OWS3 test bed.

Step 1a, caching a copy of the metadata, stores a copy of the source metadata document in a repository that is under the control of the catalog. For the **Harvest** operation, this can happen automatically since the catalogue reads a copy of the metadata document directly based on the value of the *Source* parameter. For the **Transaction** operation, a multipart document must be created where one of the parts is the source document.

7.2.1.1 Inserting repository items with a Transactions (from 05-025r3)

When inserting extrinsic objects along with one or more repository items, the multipart/form-data content type (RFC 2388) must be used. Multipart media types such as this one are intended for compound messages that consist of several interrelated parts; such entities comprise a root part plus any number of other parts or attachments. The repository items are included as additional file parts, with the **Content-Type** headers set accordingly. An XML document with the **Transaction** element as the document element must be included in the part named Transaction. That is, the following request message headers must be set for this root part:

```
Content-Disposition: form-data; name="Transaction"
Content-Type: application/xml
```

In order to relate an extrinsic object to a repository item, the value of the name parameter in the Content-Disposition header for the part containing the repository item must match the value of the id attribute for the extrinsic object in the body of the root part. While the id value must be a URN, an experimental namespace identifier may be used (e.g. x-foo) to provide a temporary reference to a message part; upon publication this value will be replaced with a UUID value generated by the catalogue. Any of the optional disposition parameters defined in RFC 2183 may also be included (e.g., filename, modification-date), but their usage is unconstrained by this profile and they may be safely ignored.

EXAMPLE

Required message headers for a message part containing a repository item.

```
Content-Disposition: form-data; name="urn:x-foo:img-1"
Content-Type: image/png
```

Support for multipart/form-data content is required this content type is supported by most HTTP user agents and client toolkits. The multipart/related media type (RFC 2387, commonly used in SOAP bindings) may also be employed, but it is optional. The catalogue shall advertise all supported content types in the mime-types service property.

8 Web map service capability documents

8.1 Introduction

The version 1.1.3 capabilities document, for a web map service, is composed of two first level elements: a *Service* element that contains metadata about the service provider and a *Capability* element that defines which WMS operations the service provides and the layers it serves. Figure 1 graphically illustrates the two first-level elements that make up a WMS capabilities document.

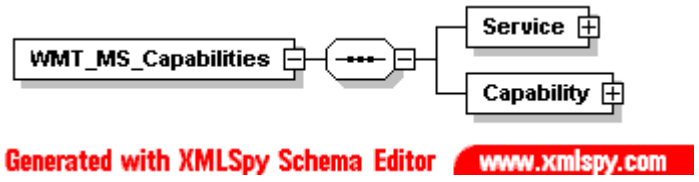


Figure 3 – WMS Capabilities Document Structure

8.2 Service Metadata

Figure 2 illustrates the elements contained in the *Service* element of a WMS capabilities document.

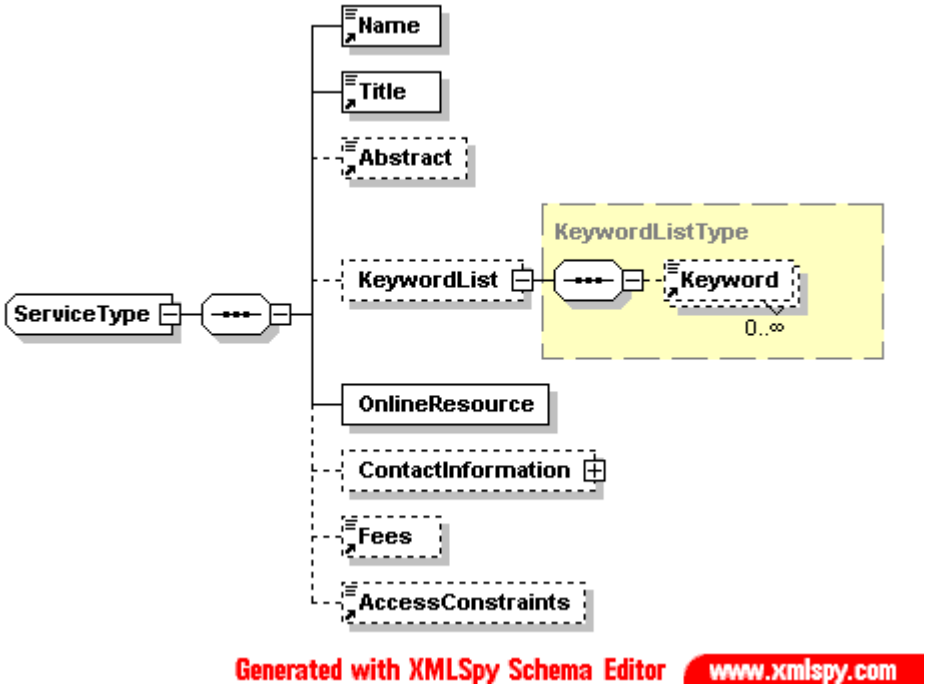


Figure 4 – Service Elements

Table 1 shows how the *Service* elements for the WMS capabilities document may be mapped into the *Service* elements of the ebRIM class *Service*.

Table 1 – Mapping WMS Service Element to ebRIM

WMS 1.1 Element Path	ebRIM Path
Service/Title	Service/Name/LocalizedString/@value
Service/Name	Service/Slot[@name="Identifier"]/ValueList/@value[1]
Service/Abstract	Service/Description/LocalizedString/@value
Service/KeywordList/Keyword	Service/Slot[@name="Keywords"]/ValueList/Value
Service/OnlineResource	Service/Slot[@name="OnlineResource"]/ValueList/Value
Service/Fees	Service/Slot[@name="Fees"]/ValueList/Value
Service/AccessConstraints	Service/Slot[@name="AccessConstraints"]/ValueList/Value

Once an instance of the class *Service* has been created in the registry, to register the WMS, the record must be classified using a service taxonomy to indicate that the service is a *Web Map Service*.

At the moment no standard service taxonomy exists in OGC. In the OWS1.2 test bed, two service taxonomies were used. The first taxonomy was the ISO19119 taxonomy that is listed in Appendix X. The second taxonomy was created to classify services as a web map service, web feature service, web coverage service or web registry service. The taxonomy was flat and contained four nodes, one for each service type. The taxonomy was registered in the registry using an instance of the *ClassificationScheme* class.

A registered server must be classified using a service taxonomy at least once but may be classified any number of times.

A new instance of the *Classification* class must be created to classify the newly registered service as a *Web Map Service*. If, for example, the simple service taxonomy described above had a unique identified of **13,** and the node in that taxonomy that indicated that a service was a web map service had a unique identifier of **76**, and the unique identifier for the newly registered web map service was **10,** then the following instance of the *Classification* class:

```
<Classification classifiedObject="10" classificationScheme="13" classificationNode="76" />
```

created in the registry would indicate that the service with unique identifier **10** is a web map service (i.e. node id of **76**) according to the classification scheme with unique identifier **13**.

The *Service* object is linked to a *ServiceBinding* instance. The *ServiceBinding* is intended to keep the access point of the *Service*. Actually, the URI of the GetCapabilities operation (in the HTTP/GET method) specified in the Capabilities document is mapped to the *accessURI* parameter of the *ServiceBinding* object.

---

[1] A *Slot* is an extensibility mechanism that exists in the ebRIM.

Note that the *Service* and *ServiceBinding* classes are doubly linked through *ServiceBinding* elements of the *Service* class and the *service* attribute of the *ServiceBinding* class (links are made by using unique identifiers).

Table 2 – Service access point into ebRIM object

WMS 1.1 Element Path	ebRIM Path
Capability/Request/*GetCapabilities Operation*/DCPType/HTTP/GET/onlineResource	ServiceBinding/@accessURI

Figure 4 shows the elements used to encode contact information in a WMS capability document.

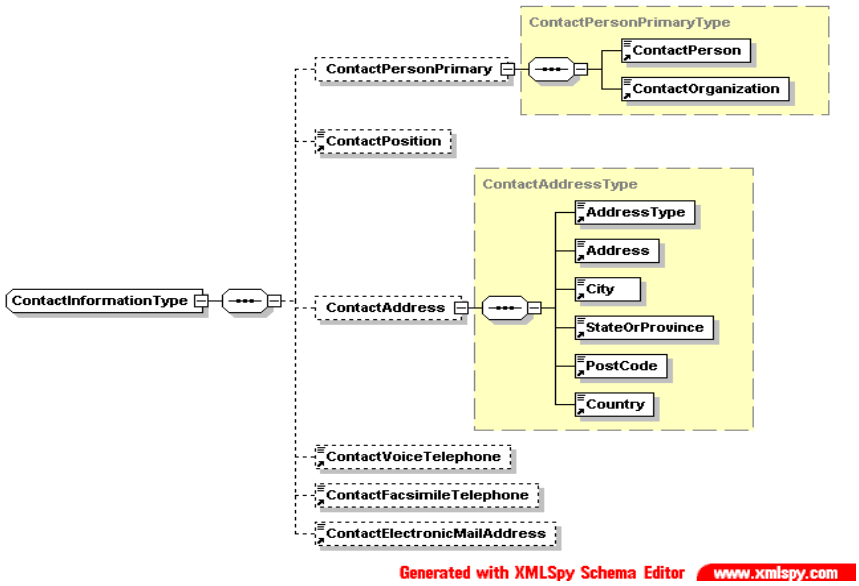


Figure 5 – Contact Information Element in WMS Capability Document

Table 3 shows how the contact information elements from a WMS capabilities document are mapped into the *User* and *Organization* classes that are part of the ebRIM.

Table 3 – Contact Information Mapping

WMS 1.1.1 Element Path	ebRIM Elements
Service/ContactInformation/ContactPrimaryPerson/ContactPerson	User/PersonName
Service/ContactInformation/ContactPrimaryPerson/Organization	Organization/Name
Service/ContactInformation/ContactPosition	User/Slot[@name="ContactPosition"]/ValueList/Value
Service/ContactInformation/ContactAddress/AddressType	User/Address/Slot[@name="AddressType"]/ValueList/Value
Service/ContactInformation/ContactAddress/Address	User/Address/@streetNumber
Service/ContactInformation/ContactAddress/City	User/Address/@city

Service/ContactInformation/ContactAddress/StateOrProvince	User/Address/@stateOrProvince
Service/ContactInformation/ContactAddress/PostCode	User/Address/@postalCode
Service/ContactInformation/ContactAddress/Country	User/Address/@country
Service/ContactInformation/ContactVoiceTelephone	User/TelephoneNumber
Service/ContactInformation/ContactFacsimileTelephone	User/TelephoneNumber
Service/ContactInformation/ContactElectronicMailAddress	User/EmailAddress

Note that *TelephoneNumber* and *EmailAddress* objects have a Slot which is intended to specify the type (or context) in which these objects are used (i.e. “Mobile” or “FacSimile” for the *TelephoneNumber*).

The *User* and *Organization* classes of the ebRIM are doubly linked through the *primaryContact* attribute of the *Organization* class and the *organization* attribute of the *User* class (i.e. the value of the *primaryContact* attribute is equal to the unique identifier of the *User* instance and the value of the *organization* attribute is equal to the unique identifier of the *Organization* instance).

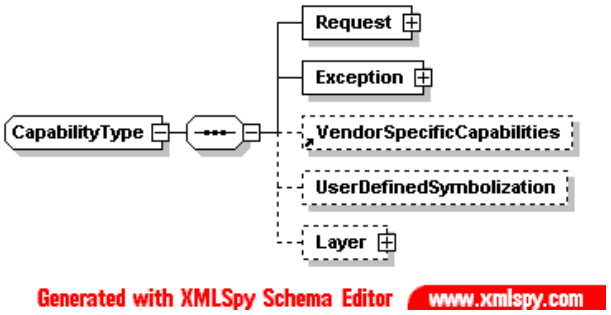
The *Organization* class and the *Service* classes of the ebRIM may be related using the *Association* class. Thus when mapping the information from a WMS capabilities document, a *Service* class instance is created in the registry and an *Organization* class instance is created in the registry and then an *Association* class instance is created to relate the entries. If, for example, the unique identifier for the *Service* instance is **10** and the unique identifier for the *Organization* instance is **145** then the following *Association* instance would need to be created to relate the two records:

```
<Association associationType=”OffersService” sourceObject=”145” targetObject=”10” />
```

This *Association* instance is saying that *Organization* with id **145** offers[2] *Service* with id **10.**

8.3 Capability Metadata

The capability metadata is used to describe which WMS operations that this particular service offers and the layers that the service offers. Figure 6 illustrates the basic structure of the *Capability* element:



[2] A partial set of association type values is shown in Annex C. This document extends some of the lists in ANNEX C by adding new object types and association types.

Figure 6 – Structure of the Capability Element

8.3.1 Request Metadata

Figure 7 shows the structure of the *Request* element.

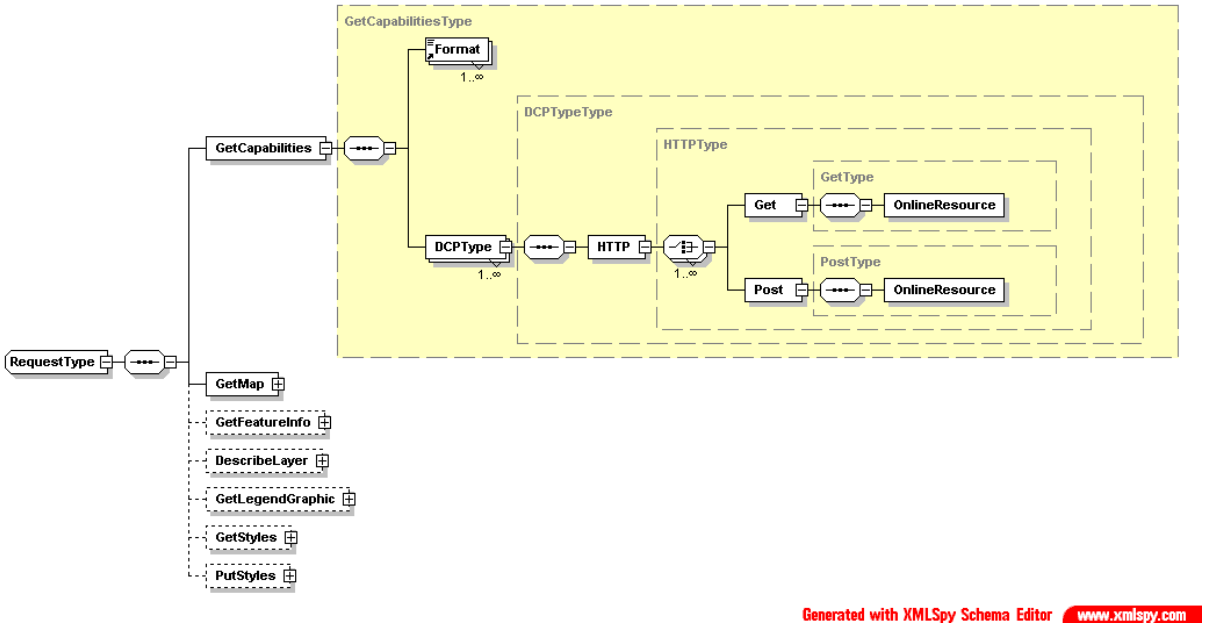


Figure 7 – Structure of the Request Element

The structure of the elements *GetMap*, *GetFeatureInfo*, *DescribeLayer*, *GetLegendGraphic*, *GetStyles* and *PutStyles* is identical to the structure of the *GetCapabilities* element shown in figure 7. In the following discussion, the value *OperationName* is a placeholder for the operation element being mapped (i.e. one of *GetMap*, *GetFeatureInfo*, *DescribeLayer*, *GetLegendGraphic*, *GetStyles* and *PutStyles*).

Table 3 shows the mapping of each **/Capability/Request/*OperationName*** element into the ebRIM.

Table 3 – Request Element Mapping

WFS 1.1 capabilities element paths	ebRIM element paths
Capability/Request/*OperationName*	ServiceBinding[@service="*id of service*"]/Name/LocalizedString/@value
Capability/Request/*OperationName*/Format	ServiceBinding[@service="*id of service*"]/Slot[@name="Format"]/ValueList/Value
Capability/Request/*OperationName*/DCPType/HTTP/Get/OnlineResource/@xlink:href	ServiceBinding[@service="*id of service*" and @accessURI="*access point URI*"] ServiceBinding[@service="*id of service*"]/Slot[@name="DCPType"]/ValueList/[Value="HTTP GET"]

Capability/Request/OperationName/DCPType/HTTP /Post/OnlineResource/@xlink:href	ServiceBinding[@service="*id of service*" and @accessURI="*access point URI*"] ServiceBinding[@service="*id of service*"]/Slot[@name="DCPType"/ValueList/[Value="HTTP POST"]
NOTE: The phrase "*access point URI*" is mean to indicate that the accessURI attribute should be assigned the value of the xlink:href attribute from the source document	

Each service binding is associated to its service via the **service** attribute on the **ServiceBinding** element.

The **specificationLink** attribute for each service binding shall point to an **ExternalLink** record that inturn points to the WMS specification in the OGC document repository (http://portal.opengeospatial.org/files/?artifact_id=5316).

8.3.2 Exception Metadata

Figure 8 shows the structure of the *Exception* element.

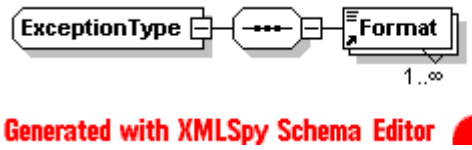


Figure 8 – Exception element structure

The exception format information may be mapped into the *Service* class created in clause 6.2 as a *Slot*. A *Slot* is an extensibility mechanism that the ebRIM provides.

Table 4 shows the mapping.

Table 4 – Exception Element Mapping

WMS 1.1.1 Element Path	ebRIM Element Path
Capability/Exception/Format	Service/Slot[@name=”ExceptionFormats”]/ValueList/Value

Issue 1. Note from Renato to Peter : we should maybe consider the storing of the ISO19119 document into an ExtrinsicObject linked to the Service object. RESPONSE: I think all metadata documents describing any resource (service or not) should be registered as ExtrinsicObject of type ISO19119 (or whatever the metadata format is) and then associated to the resource via an Association with the association type set to "DescribedBy" (or something like that). So, in

the case of a service we would have the association "<<Service>> *DescribeBy* <<ISO19119>>".

8.3.3 Layer Metadata

In this section we will deal with mapping the layer metadata. Figure 8 shows the structure of the layer metadata.

Table 5 shows the mapping of the layer metadata element into the ebRIM. Layer metadata is mapped into the RIM using an instance of the *ExtrinsicObject* class where the *objectType* is set to **Data_Set**.

Table 5 – Layer Element Mapping

WMS 1.1 Element Path	ebRIM Element Path
Capability/Layer/Title	ExtrinsicObject[@objectType=”Data_Set”]/Name/LocalizedString/@value
Capability/Layer/Name	ExtrinsicObject[@objectType=”Data_Set”]/Slot[@name=”Identifier”]/ValueList/Value
Capability/Layer/Abstract	ExtrinsicObject[@objectType=”Data_Set”]/Description/LocalizedString/@value
Capability/Layer/KeywordList/Keyword	ExtrinsicObject[@objectType=”Data_Set”]/Slot[@name=”Keywords”]/ValueList/Value
Capability/Layer/SRS	ExtrinsicObject[@objectType=”Data_Set”]/Slot[@name=”SRS”]/ValueList/Value
Capability/Layer/LatLonBoundingBox	ExtrinsicObject[@objectType=” Data_Set”]/ Slot[@name=”Footprint”]/ValueList/Value
Capability/Layer/BoundingBox	ExtrinsicObject[@object_Type=”Data_Set”]/ Slot[@name=”Native_Footprint”]/ValueList/Value

Geometries are stored into *Slots* of the *ExtrinsicObject* “Data_Set”. They are encoded in GML in such way that XML Slot encoding looks like :

```
<ebxml:Slot name="Native_FootPrint" slotType="Geometry">
 <ebxml:ValueList>
  <ebxml:Value xmlns:gml="http://www.opengis.net/gml">
   <gml:Polygon srsName="EPSG:26986">
    <gml:outerBoundaryIs>
     <gml:LinearRing>
      <gml:coordinates>227317.3811199999,889948.2662399999
238669.28896000012,889948.2662399999 238669.28896000012,901300.1740800001
227317.3811199999,901300.1740800001 227317.3811199999,889948.2662399999</gml:coordinates>
     </gml:LinearRing>
    </gml:outerBoundaryIs>
   </gml:Polygon>
  </ebxml:Value>
 </ebxml:ValueList>
</ebxml:Slot>

<ebxml:Slot name="FootPrint" slotType="Geometry">
 <ebxml:ValueList>
  <ebxml:Value xmlns:gml="http://www.opengis.net/gml">
   <gml:Polygon srsName="EPSG:4326">
    <gml:outerBoundaryIs>
     <gml:LinearRing>
      <gml:coordinates>-71.1689210448366,42.259045830448 -
71.0305843328502,42.259045830448 -71.0305843328502,42.3617240104175 -
71.1689210448366,42.3617240104175 -71.1689210448366,42.259045830448</gml:coordinates>
     </gml:LinearRing>
```

```
      </gml:outerBoundaryIs>
     </gml:Polygon>
    </ebxml:Value>
   </ebxml:ValueList>
  </ebxml:Slot>
```

A characteristic of the layer metadata in a WMS capabilities document is that the layers may be nested one or more levels deep. This nested structure is encoded in the registry by, again, using the *Association* class with the *associationType* attribute set to **Contains.** For example, consider the following structure of nested layers:

```
Layer1
   Layer2
   Layer3
      Layer4
   Layer5
```

The following set of associations could be used to register this structure:

```
<Association associationType="Contains" sourceObject="1" targetObject="2" />
<Association associationType="Contains" sourceObject="1" targetObject="3" />
<Association associationType="Contains" sourceObject="3" targetObject="4" />
<Association associationType="Contains" sourceObject="1" targetObject="5" />
```

Issue 2. Note from Renato to Peter : the hierarchy in layers definition is only there to allow some layers inherits properties and values defined in more general layers. But we can keep the hierarchy in the ebRIM model if it is really needed … RESPONSE: I think maintaining it if possible is a good thing since there might be an semantic associated with the hierarchy – e.g. layers grouped into themes.

The *MetadataURL* element is used to point to detailed standardized metadata. Such metadata would be registered as a separate object, using an instance of the *ExtrinsicObject* class with the *objectType* attribute set to **ISO19139.** As instance of the *Association* class with the *associationType* attribute set to "HasForMetadata" would then be used to link the standard metadata with the layer.

This Metadata URL is stored in a Slot named "Content" in this way :

```
<ebxml:Slot name="Content" slotType="URI">
 <ebxml:ValueList>
  <ebxml:Value>
http://rp:8080/ionicwrs/wrs/WRS?request=GetExtrinsicContent&amp;service=WRS&amp;version=2
.0.0&amp;id=urn%3Auuid%3A6ba25e94-62e1-4a25-afb1-e0cae48fb671</ebxml:Value>
 </ebxml:ValueList>
</ebxml:Slot>
```

Issue 3. Note from Renato to Peter : I think "Dataset_Description" should be split into "ISO19139", "FGDC", … RESPONSE: Not sure about

that. I would either encode that type of information into a slot or use a classification to indicate the specified metadata document type.

Issue 4. Note from Renato to Peter : we should define the behaviour of the Catalog and the impact onto the data model when we update an already registered WMS... RESPONSE: I agree but we did not really discuss this during OWS3. Before we add anything we should investigate how each of us handles re-harvesting and see if a consensus approach emerges. For example, does the Ionic catalog delete the old information and replace it with the reharvested information OR does it version the information?

8.4 Mapping represented in UML

First, we will only draw the Service object and its slots:

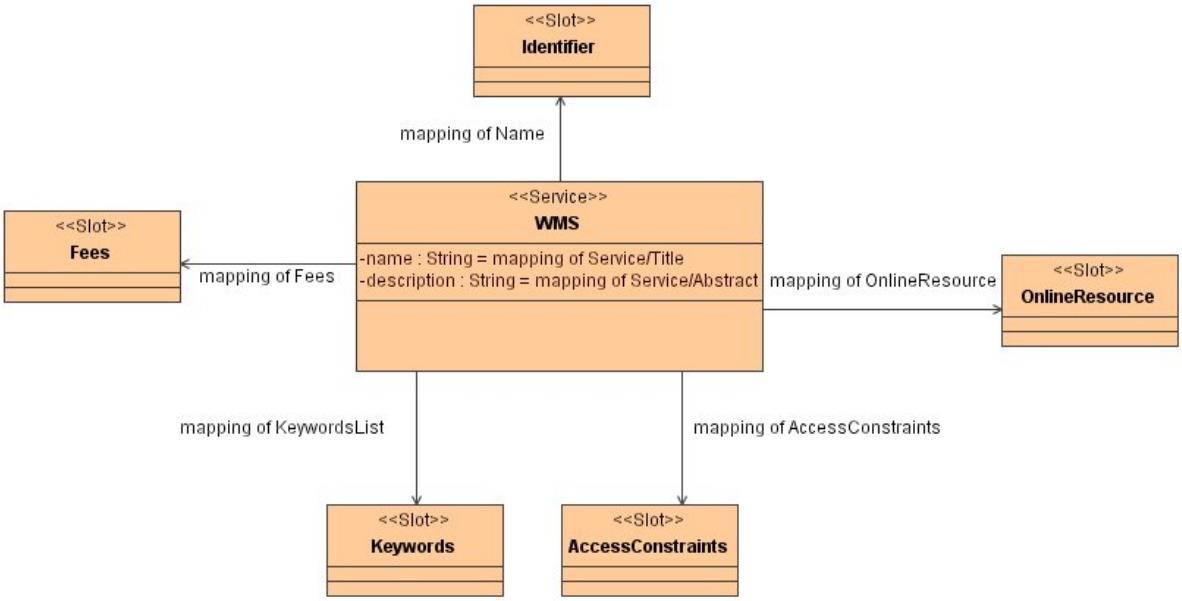


Figure 9 – WMS Service Metadata

We will now consider the mapping of the contact information:

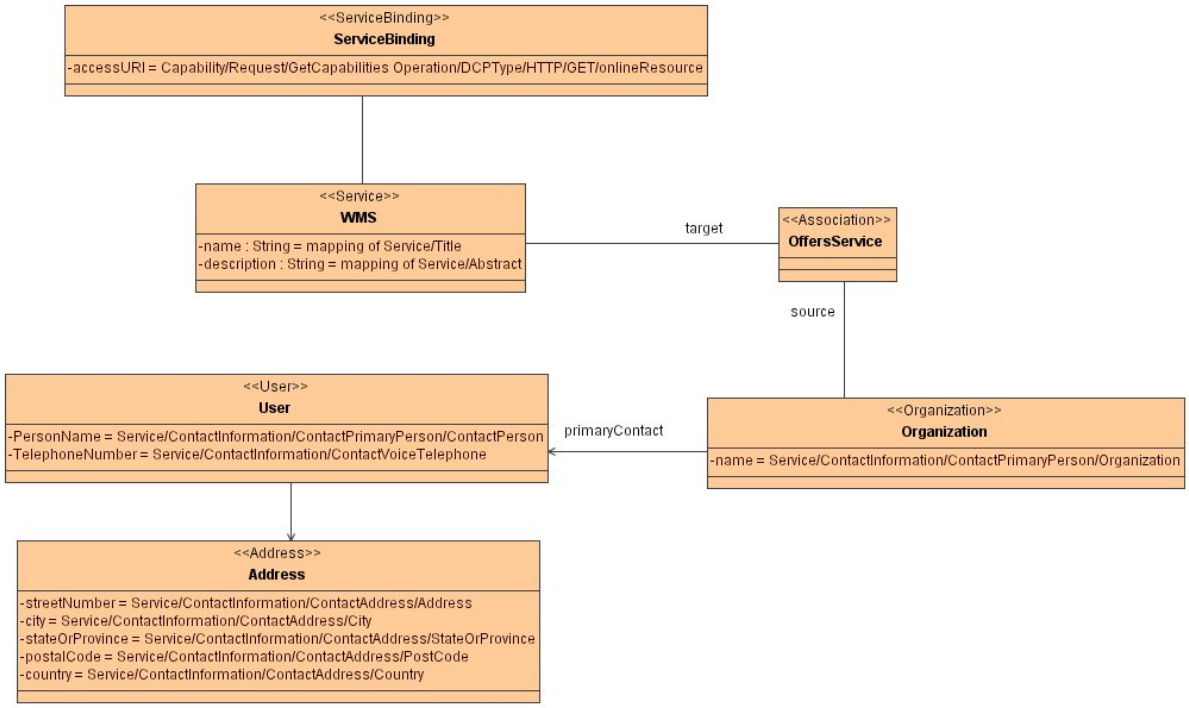


Figure 10 – WMS Contact Information

Now, let's see the relationship between the service and its layers:

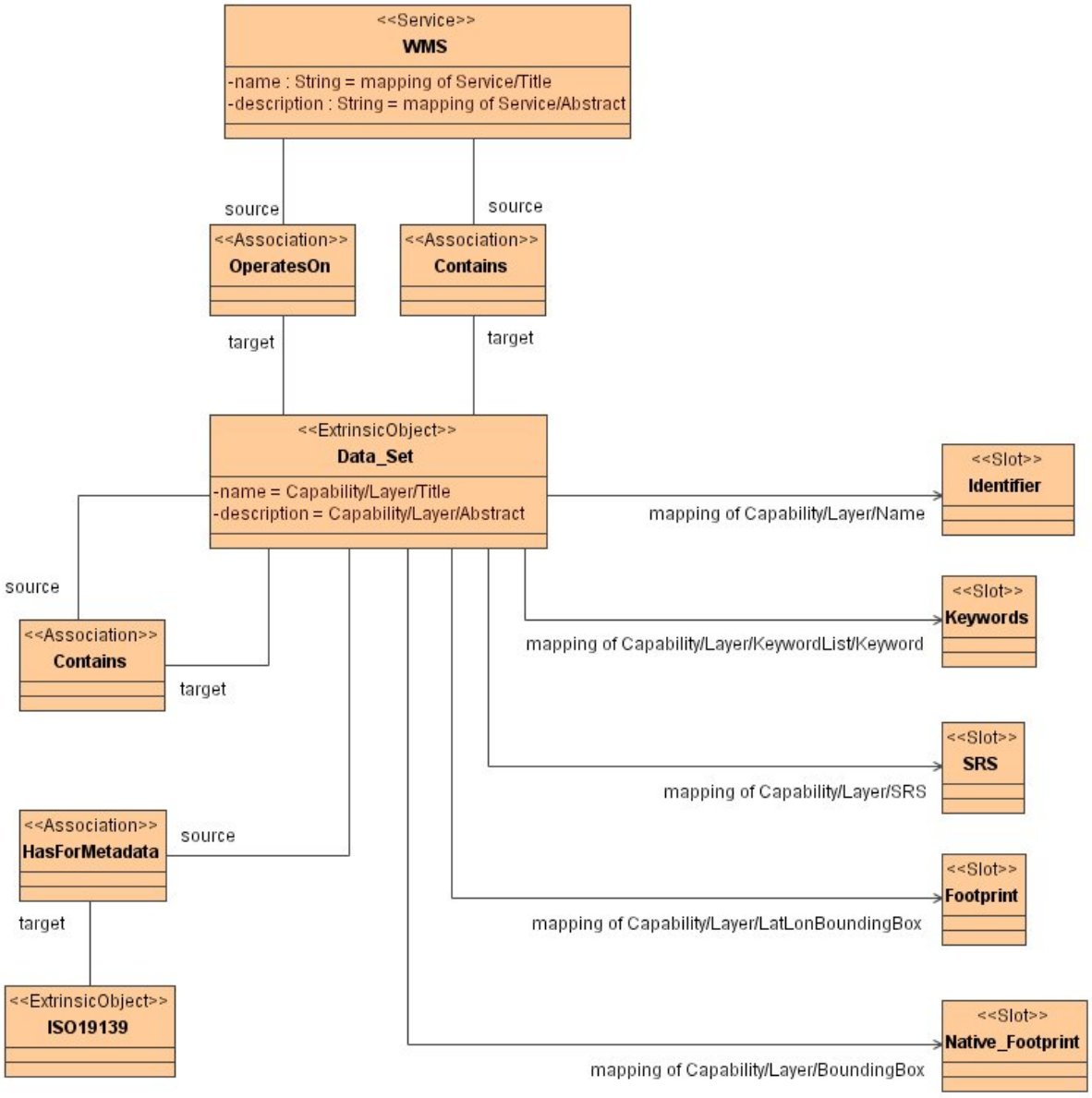


Figure 11 – Layer Metadata

9 Web feature service capability document

9.1 Introduction

The WFS V1.1 capabilities document is based on the OWS Common specification [X] and is composed of five sections: the service identification, the service provider, operations metadata, feature type list and filter capabilities. The service identification metadata describes what service is being provides and any fees or constraints that the service imposes. The service provider section contains metadata about the entity offering the service. The

operations metadata describes which operations from the WFS V1.1 specification the service supports and parameters domains and constraints on each operations. The feature type list contains a list of feature types that the service offers and provides some very lightweight metadata about each layer including bounding boxes and supported coordinate reference systems.

It should be noted that much of the information in the clause may be used to map the capability documents of other OGC services that user the OGC Common specification as the basis of their capability document.

9.2 Service Identification Metadata

The service identification metadata describes the service being offered and any fees or constraints of that service. The information in this section maps into the ebRIM *Service* entity as shown in the table 6 below.

Table 6 – Mapping ServiceIdentification elements to ebRIM

WFS 1.1 Capabilities Element Paths	ebRIM Element Paths
ServiceIdentification/ServiceType	Record classified according to an OGC service taxonomy.
ServiceIdentification/ServiceTypeVersion	Service/Slot[@name="Version"]/ValueList/@value
ServiceIdentification/Title	Service/Slot[@name="Title"]/ValueList/Value
ServiceIdentification/Abstract	Service/Description/LocalizedString/@value
ServiceIdentification/Name	Service/Name/LocalizedString/@value
ServiceIdentification/Keywords/Keyword	Service/Slot[@name="Keywords"]/ValueList/@value
ServiceIdentification/Keywords/Type	Service/Slot[@name="KeywordAuth"]/ValueList/@value
ServiceIdentification/Fees	Service/Slot[@name="Fees"]/ValueList/@value
ServiceIdentification/AccessConstraints	Service/Slot[@name="AccessConstraints"]/ValueList/@value

The ServiceType element, from the WFS capabilities document, does not need to be mapped since ebRIM has a built-in classification mechanism and the service record will simply be classified according to the OGC service taxonomy. Of course, the service may be classified using any number of service taxonomies that are available in an ebRIM registry. This document proposes that an OGC catalogue contain at least the OGC service taxonomy and the ISO19119 service taxonomy and that OGC web service be classified using (at least) these two taxonomies.

A new instance of a *Classification* record must be created each time a service is classified. The instance of the *Classification* record will contain the id of the service records and the id of the classification node that classifies the record. As proposed, a newly registered service will create at least two instances of the *Classification* records. One for the OGC service taxonomy and one for the ISO19119 service taxonomy.

Most of the other service identification elements map directory in ebRIM as slots except *Abstract* that maps directly to the *Description* element in ebRIM, and *Title* that maps directly to the *Name* element in ebRIM.

9.3 Service Provider Metadata

The service provider information identifies the entity that is offering the service.

Table 7 – Mapping ServiceProvider elements to ebRIM

WFS 1.1 capabilities element path	ebRIM element path
…/ProviderName	Organization/Name/LocalizedString/@value
…/ProviderSite/@xlink:href	ExternalLink/@externalURI
…/ServiceContact/IndividualName	User/PersonName
…/ServiceContact/PositionName	User/Slot[@name=”Position”]/ValueList/…
…/ServiceContact/ContactInfo/Phone/Voice	User/TelephoneNumber
…/ServiceContact/ContactInfo/Phone/Facsimile	User/TelephoneNumber
…/ServiceContact/ContactInfo/Address/DeliveryPoint	User/Address/@streetNumber
…/ServiceContact/ContactInfo/Address/City	User/Address/@city
…/ServiceContact/ContactInfo/Address/AdministrativeArea	User/Address/@stateOrProvince
…/ServiceContact/ContactInfo/Address/PostalCode	User/Address/@postalCode
…/ServiceContact/ContactInfo/Address/Country	User/Address/@country
…/ServiceContact/ContactInfo/Address/ElectronicMailAddress	User/EmailAddress
…/ServiceContext/ContactInfo/OnlineResource/@xlink:href	ExternaLink/@externalURI
…/ServiceContext/ContactInfo/ContactInstructions	User/Slot[@name=”ContactInstruction”]/ValueList /Value
…/ServiceContact/Role	User/Slot[@name=”Role”]/ValueList/Value

The *User* and *Organization* classes of the ebRIM are linked through the **primaryContact** attribute of the *Organization* class. The value of the **primaryContact** attribute is equal to the unique identifier of the *User* instance.

Similarly, the *ExternalLink* instances are linked to the *Organization* and *User* classes using the *Association* class. The *AssocitationType* name shall be **HasUrl**.

9.4 Operations Metadata

The following table shows how to map the operations supported by a service using the ServiceBinding object.

Table 8 – Mapping OperationsMetadata element to ebRIM

WFS 1.1 capabilities element paths	ebRIM element paths
OperationsMetadata/Operation/@name	ServiceBinding[@service=*"id of service"*]/Name/LocalizedString/@value
OperationsMetadata/Operation/DCP/HTTP/Get/@xlink:href	ServiceBinding[@service="*id of service* and @accessURI="*access point URI"]* ServiceBinding[@service="*id of service*"]/Slot[@name="DCPType"/ValueList/[Value="HTTP GET"]
OperationsMetadata/Operation/DCP/HTTP/Post/@xlink:href	ServiceBinding[@service="*id of service*" and @accessURI="*access point URI"*] ServiceBinding[@service="*id of service*"]/Slot[@name="DCPType"/ValueList/[Value="HTTP

	POST"]
OperationsMetadata/Operation/DCP/HTTP/Post/InputFormat	ServiceBinding[@service="*id of service*"]/Slot[@name="InputFormat"]
OperationsMetadata/Operation/DCP/HTTP/Soap/@xlink:href	ServiceBinding[@service="*id of service*" and @accessURI="*access point URI*"] ServiceBinding[@service="*id of service*"]/Slot[@name="DCPType"/ValueList/[Value="HTTP SOAP"]
OperationsMetadata/Parameter/@name OperationsMetadata/Parameter/Value	ServiceBinding[@service="*id of service*"]/Slot[@name="*parameter name*" and @slotType="Parameter"]/ValueList/@value
OperationsMetadata/Parameter/Metadata/@xlink:href	N/A
NOTE: The phrase "*access point URI*" is mean to indicate that the **accessURI** attribute should be assigned the value of the **xlink:href** attribute from the source document.	

The service binding is linked to the service through the use of the **service** attribute on the **ServiceBinding** element.

The **specificationLink** attribute for each service binding shall point to an **ExternalLink** record that inturn points to the WFS 1.1 specification in the OGC document repository (https://portal.opengeospatial.org/files/?artifact_id=8339).

9.5 FeatureTypeList Metadata

The following table maps the metadata for each feature type offered by a service into the ebRIM.

Table 9 – Mapping of FeatureType Metadata to ebRIM

WFS 1.1 Element Path	ebRIM Element Path
…/FeatureType/Title	ExtrinsicObject/[@objectType=" FeatureType"]/Slot[@name="Title"]/ ValueList/Value
…/FeatureType/Name	ExtrinsicObject/[@objectType="FeatureType"]/Name/ LocalizedString[@value="…"]
…/FeatureType/Abstract	ExtrinsicObject/[@objectType=" FeatureType"]/Description/ LocalizedString[@value="…"]
…/FeatureType/KeywordList/Keyword	ExtrinsicObject/[@objectType=" FeatureType"]/Slot[@name="Keywords"]/ ValueList/Value
…/FeatureType/Operations/Operation	ExtrinsicObject/[@objectType=" FeatureType"]/Slot[@name="Operations"]/ ValueList/Value
…/FeatureType/OtherSRS	ExtrinsicObject/[@objectType=" FeatureType"]/Slot[@name=OtherSRS]/ ValueList/Value
…/FeatureType/WGS84BoundingBox	ExtrinsicObject/[@objectType=" FeatureType"]/Slot [@name= "Footprint"]/ValueList/Value
…/FeatureType/BoundingBox	ExtrinsicObject/[@objectType="FeatureType"] /Slot[@name="Native_Footprint"]/ValueList/Value

9.6 Mapping represented in UML

First, the WFS object itself and its slots:

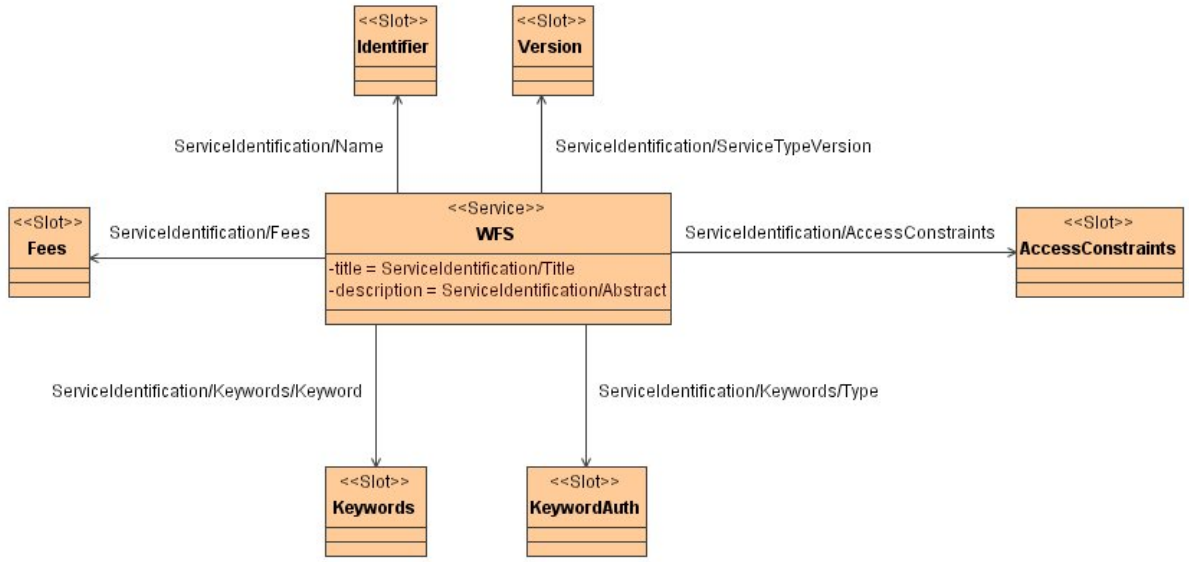


Figure 12 – WFS Service Metadata

Let's now consider the contact information:

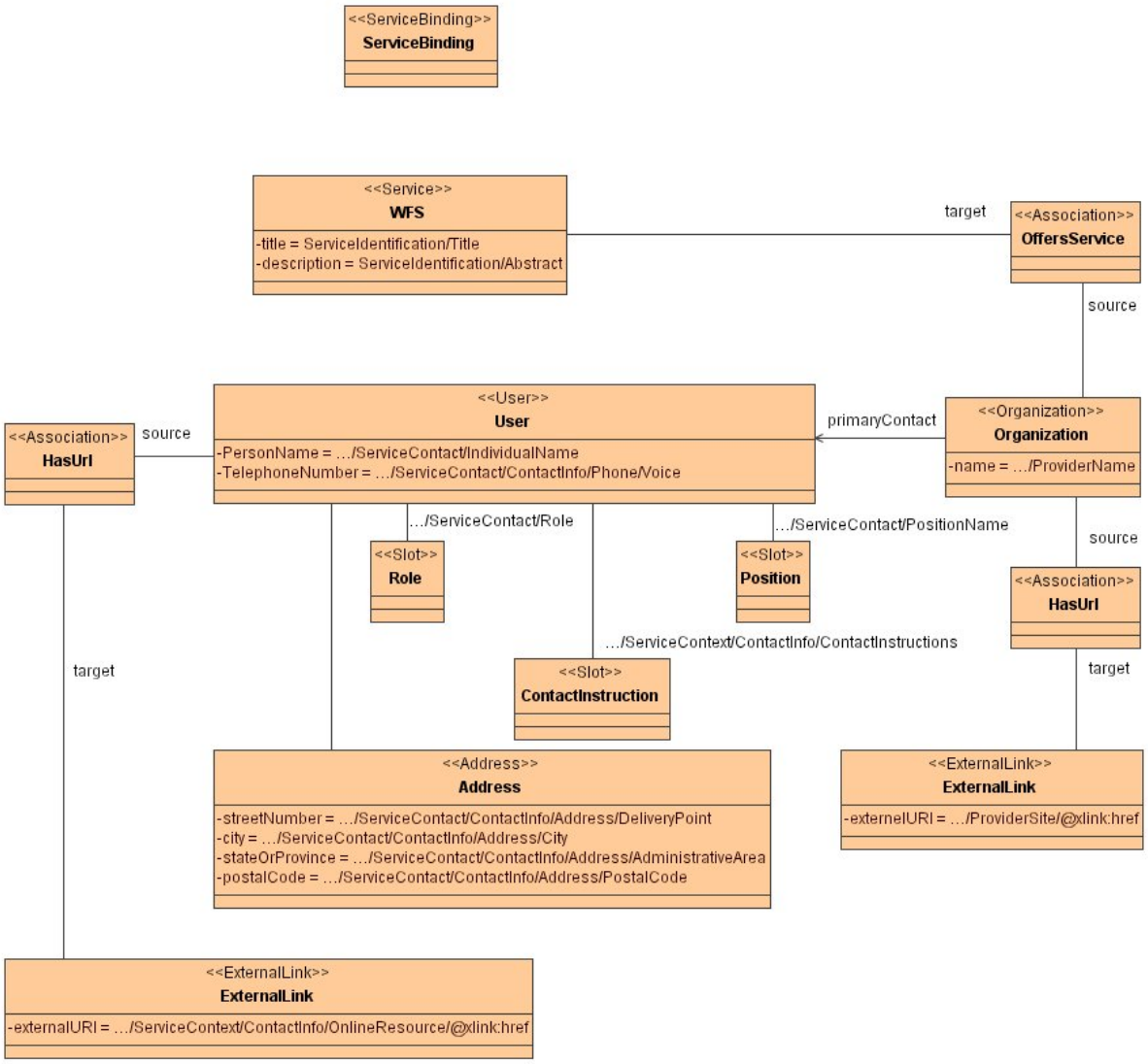


Figure 13 – WFS / ebRIM UML Model

and finally the feature types:

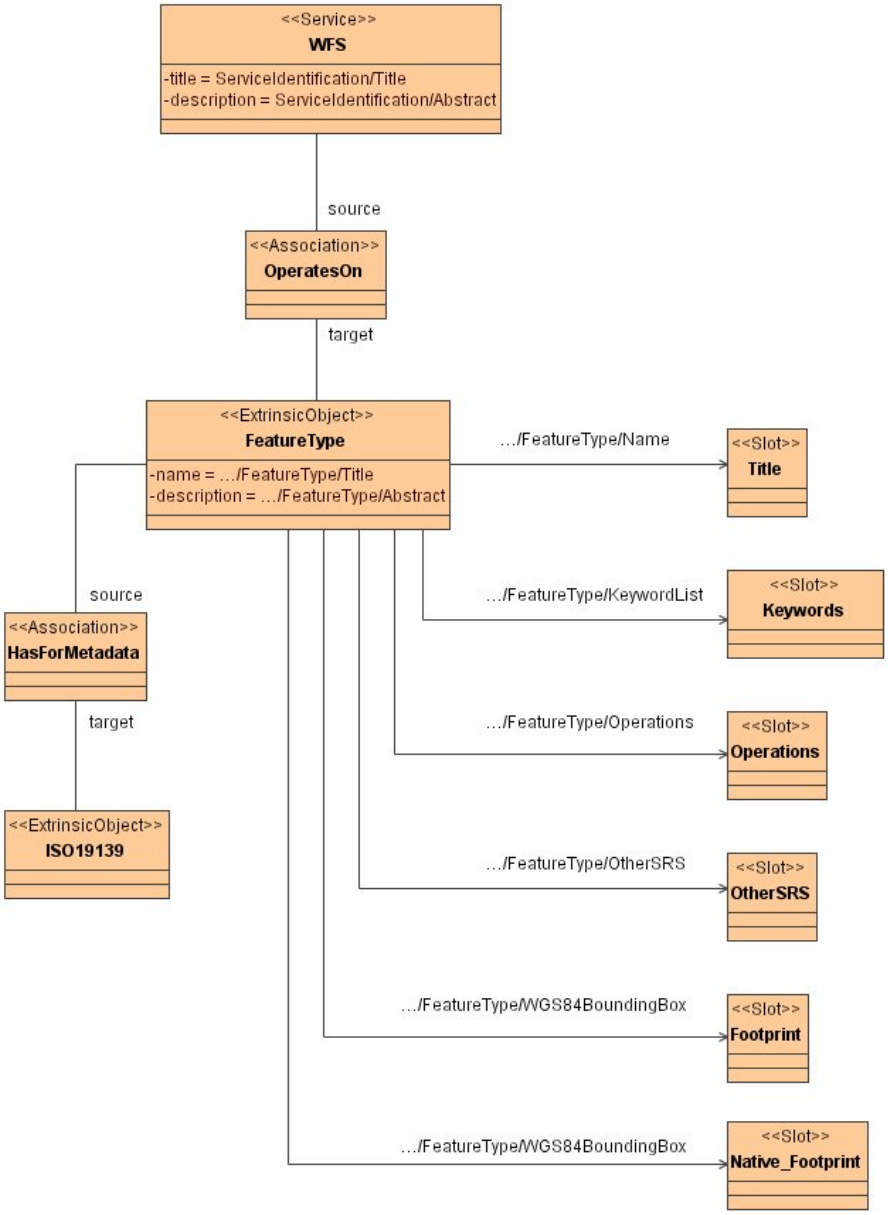


Figure 14 – WFS Operation ebRIM Mapping

10 Web coverage service capability document

10.1 Introduction

The WCS V1.0 capabilities document is based on the OWS Common specification [X] and is composed of three sections: Service, Capability and ContentMetadata.

The first section, **Service**, contains service metadata elements shared by other OGC Web Services, that provide a minimal, human readable description of the service. This section is structured much like the WMS and WFS service descriptions (name, label, keywords, fees, etc).

The second section, **Capability**, of type **WCSCapabilityType**, also resembles that for WMS and WFS. It describes the requests that the service supports, the formats in which exceptions are returned, and any other vendor-specific service capabilities.

The third section, ContentMetadata has an optional and repeatable sub-element, **CoverageOfferingBrief**, and provides detailed information about a server's coverage offerings, like name, label, description, keywords and a box.

10.2 Service

The service metadata describes the service being offered and any fees or constraints of that service. The information in this section (figure X) maps into the ebRIM *Service* entity as shown in the table Y below.

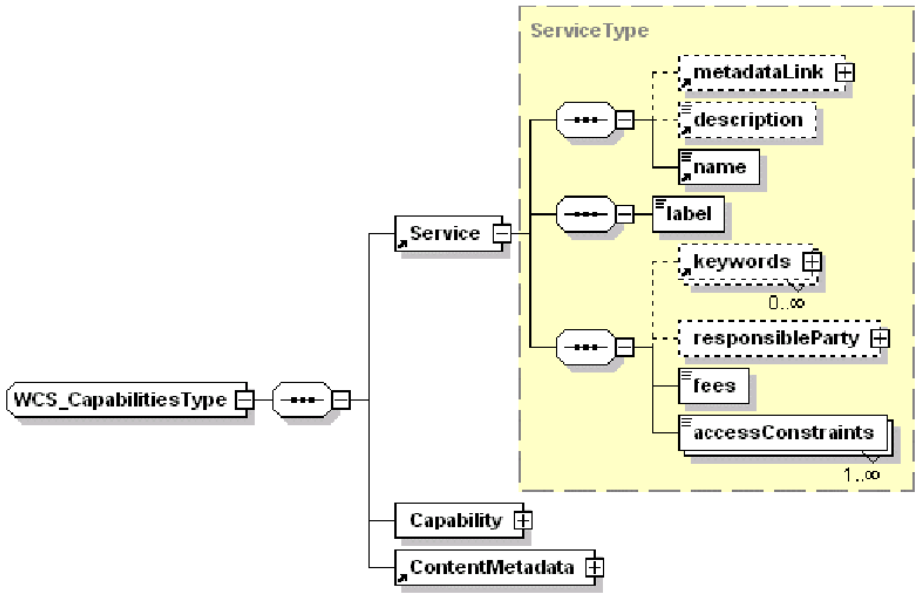


Figure 15 – xml structure of the 'Service' section of WCS 1.0

Table 10 – Mapping ServiceIdentification elements to ebRIM

WCS 1.0 Capabilities Element Paths	ebRIM Element Paths
Service/Title	Service/Name/localizedString/@value
Service/Abstract	Service/Description/LocalizedString/@value
Service/Name	Service/Slot[@name="Identifier"]/ValueList/Value
Service/Keywords/Keyword	Service/Slot[@name="Keywords"]/ValueList/@value
Service/Keywords/Type	Service/Slot[@name="KeywordAuth"]/ValueList/@value

Service/Fees	Service/Slot[@name="Fees"]/ValueList/@value
Service/AccessConstraints	Service/Slot[@name="AccessConstraints"]/ValueList/@value

The *metadataLink* element is used to point to detailed standardized metadata. Such metadata would be registered as a separate object, using an instance of the *ExtrinsicObject* class with the *objectType* attribute set to **ISO19139** (or any other supported MD format). As instance of the *Association* class with the *associationType* attribute set to "HasForMetadata" would then be used to link the standard metadata with the service.

ResponsibleParty is made of optional individualName, organizationName and positionName, in order to be one or both of individual or organization. We will associate the service with an optional User respresenting the ResponsibleParty/individualName through an associationType="HasResponsibleIndividual". We will also associate the service with an optional Organization representing the ResponsibleParty/organizationName through an associationType="HasResponsibleOrganization".

10.3 Capability

One way to map the operations is through an ExtrinsicObject of objectType="Operation", and linked to the service through an Association of associationType="SupportsOperation". The name of the extrinsicObject becomes the one of the operation (i.e., DescribeCoverage, etc). The DCPType information (Get and or Post) can be mapped in two slots name="supportsGet" and name="supportsPost", which can store only two possible values, "true" and "false".

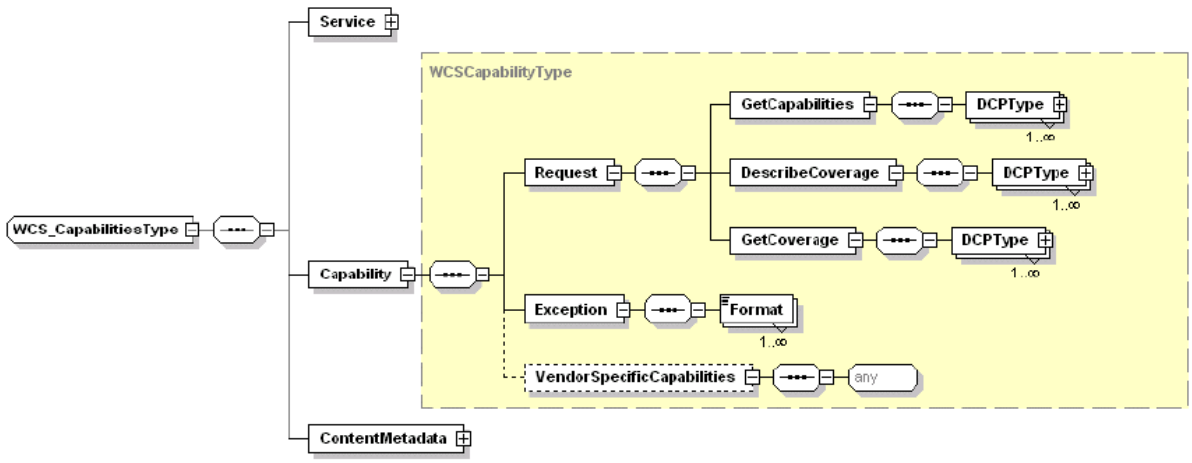


Figure 16 – xml structure of the 'Capability' section of WCS 1.0

10.4 ContentMetadata

This section is made of multiple CoverageOfferingBrief sub-sections. Each CoverageOfferingBrief will be represented with one ExtrinsicObject of objectType=

"WCS_Coverage", and linked to its containing service through an Association of associationType = "OperatesOn".

Table Y represents the mapping between one CoverageOfferingBrief and such an ExtrinsicObject.

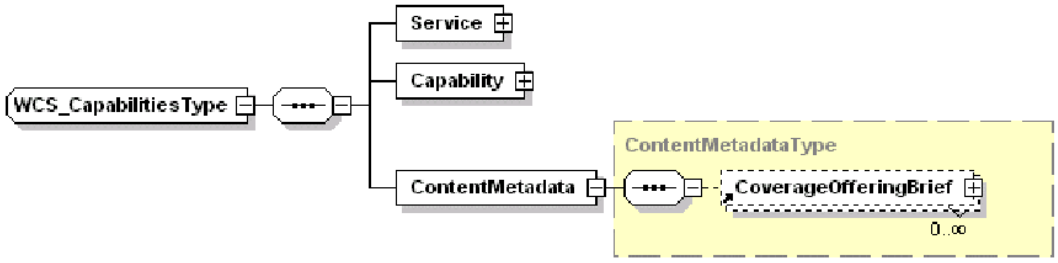


Figure 17 – xml structure of the 'ContentMetadata' section of WCS 1.0

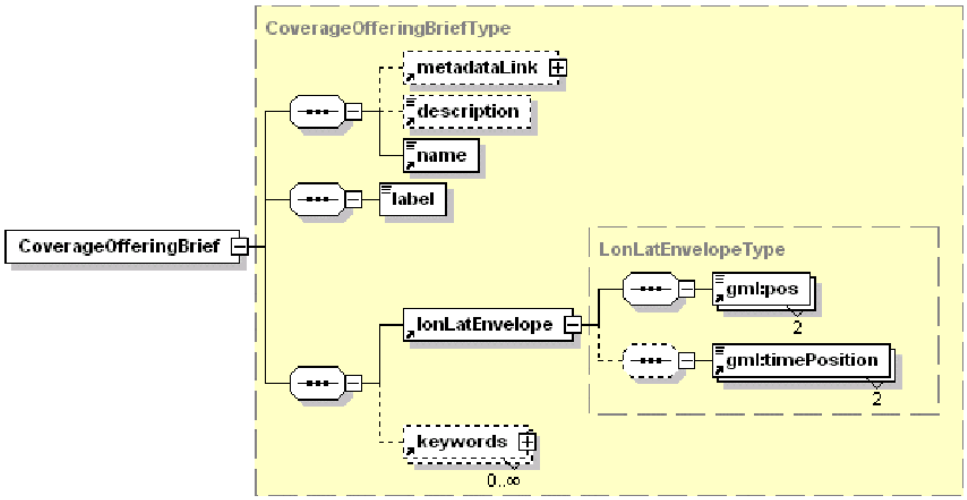


Figure 18 - xml structure of the 'CoverageOfferingBrief' section of WCS 1.0

Table 11 – CoverageOfferingBrief Element Mapping

WCS 1.0 Element Path	ebRIM Element Path
../CoverageOfferingBrief/label	ExtrinsicObject[@objectType="WCS_Coverage"]/Name/LocalizedString/@value
../CoverageOfferingBrief /Name	ExtrinsicObject[@objectType=" WCS_Coverage"]/Slot[@name="Identifier"]/ValueList/Value
../CoverageOfferingBrief /Description	ExtrinsicObject[@objectType=" WCS_Coverage"]/Description/LocalizedString/@value
../CoverageOfferingBrief/Keyword	ExtrinsicObject[@objectType=" WCS_Coverage"]/Slot[@name="Keywords"]/ValueList/Value
../CoverageOfferingBrief /lonLatEnvelope/gml:pos	ExtrinsicObject[@objectType=" WCS_Coverage"]/Slot[@name="Footprint"]/ValueList/Value
../CoverageOfferingBrief /lonLatEnvelope/gml:pos	ExtrinsicObject[@object_Type="WCS_Coverage"]/Slot[@name="Native_Footprint"]/ ValueList/Value

../CoverageOfferingBrief /lonLatEnvelope/gml:timePosition[0]	ExtrinsicObject[@object_Type="WCS_Coverage"]/Slot[@name="BeginTime"]/ValueList/Value
../CoverageOfferingBrief /lonLatEnvelope/gml:timePosition[1]	ExtrinsicObject[@object_Type="WCS_Coverage"]/Slot[@name="EndTime"]/ValueList/Value

The *metadataLink* element is used to point to detailed standardized metadata. Such metadata would be registered as a separate object, using an instance of the *ExtrinsicObject* class with the *objectType* attribute set to **ISO19139** (or any other supported MD format). As instance of the *Association* class with the *associationType* attribute set to "HasForMetadata" would then be used to link the standard metadata with this WCS_Coverage..

10.5 Mapping represented in UML

First, the WFS object itself and its slots:

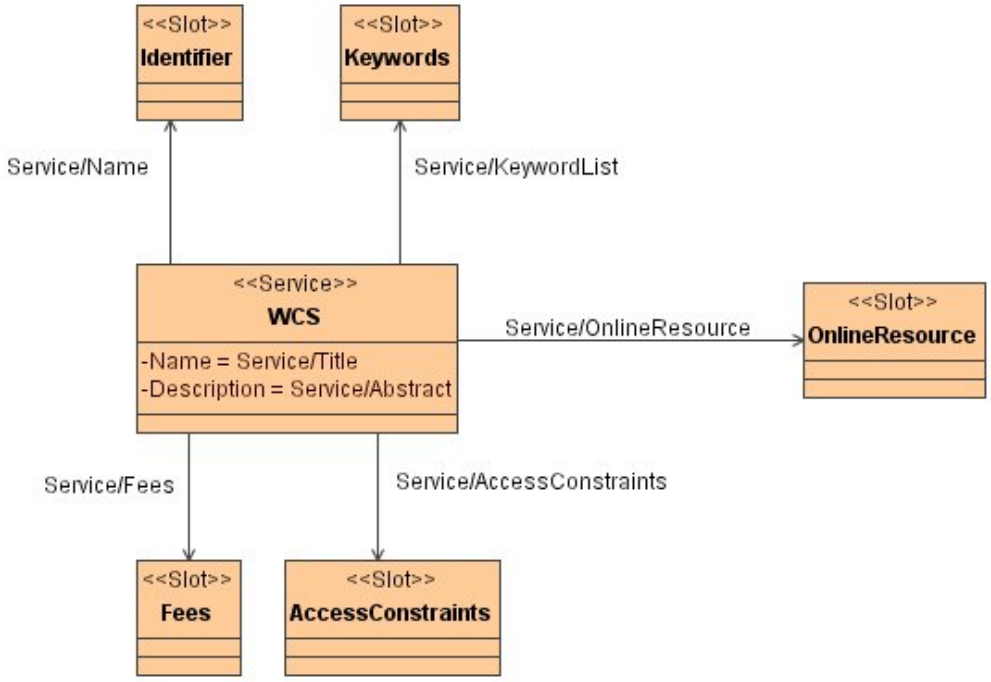


Figure 17 – WCS Service Metadata Mapping

Finally, let's represent the CoverageOfferings:

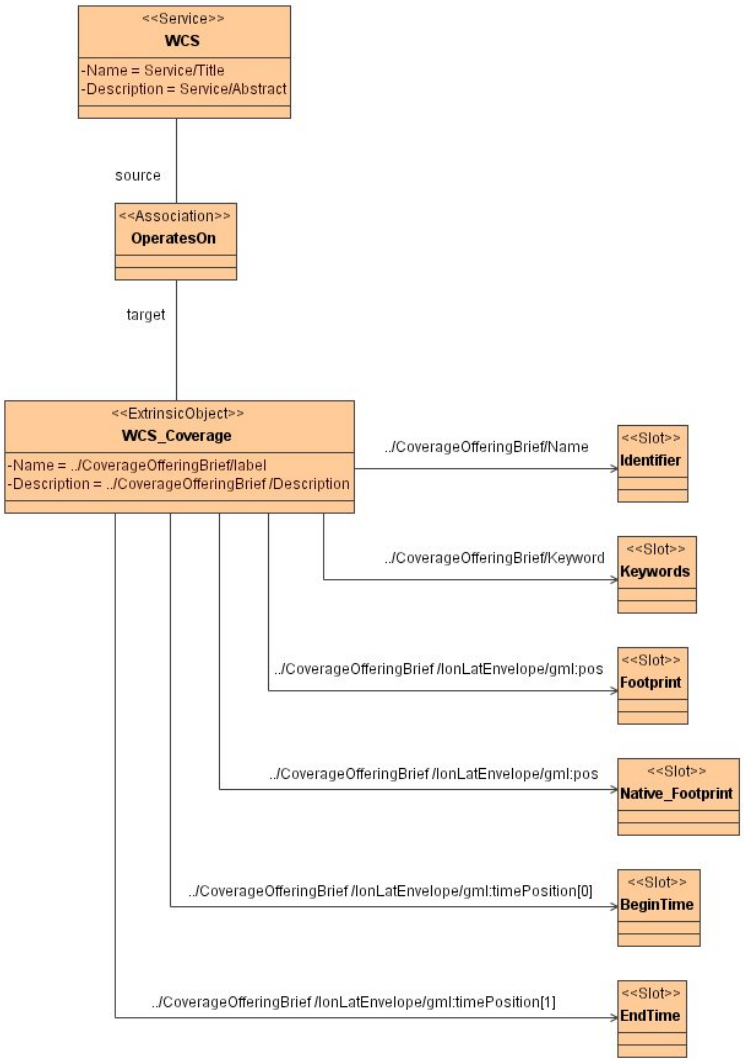


Figure 18 – Coverage Metadata Mapping

11 XML/UML schemas

In OWS3 the catalog was used as an XML schema and UML model repository. More importantly, the catalog was used to maintain the relationships between XML schemas, UML schemas, the resources declared therein and the metadata used to describe those resources. The following UML diagram illustrates these relationships as they were maintained in the catalog:

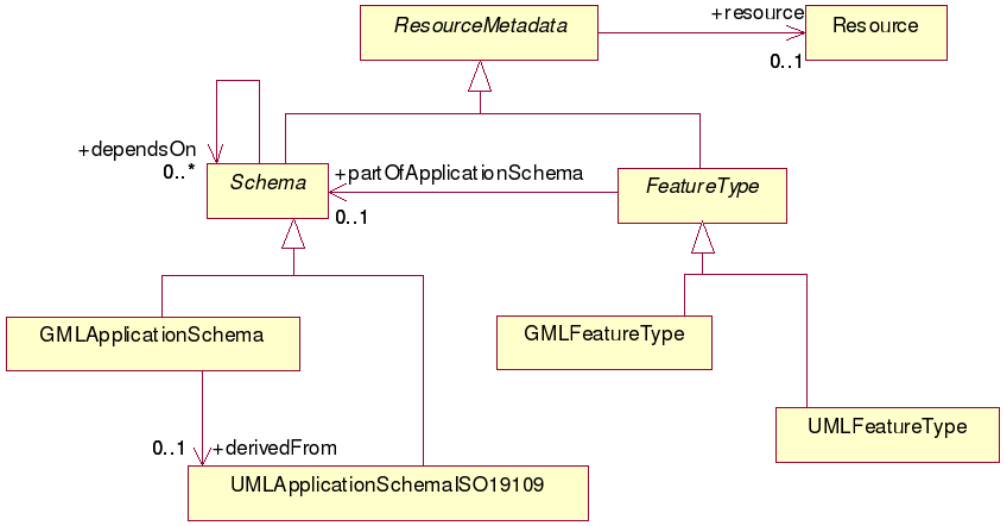


Figure 19 – XML/UML Schema Assocations

The Association entity on the ebRIM was used in the catalog to store the associations described in figure 19. The following table lists the association types used and provides a brief narrative describing the source, association type and target of each association. In the catalog these tokens were mapped to the **sourceObject**, **assocationType** and **targetObject** attributes on the **Assocation** element.

Table 12 – XML/UML Schema Associations

Association Type	Usage
DerivedFrom	• GML Schema resource **derivedFrom** UML Schema resource (created by UGAS tool)
DependsOn	• GML Schema resource ***dependsOn*** GML Schema resource (XML Schema imports) • UML Schema resource ***dependsOn*** UML Schema resource (UML dependency relationsships between packages representing application schemas)
Resource	• DDMS metadata describes **resource** resource (this is a pointer from the metadata description to the resource being described)
PartOfApplicationSchema	• GML FeatureType resource *partOfApplicationSchema* GML Schema resource • UML FeatureType resource *partOfApplicationSchema* UML Schema resource

The Department of Defense Discovery Metadata Specification version 1.2 was used as the metadata format for describing resources. This metadata was embedded in an appinfo annotation in the XML schema document where the resources being described where declared. The following XML schema fragment defines the format of the embedded metadata:

```
<xs:element name="relation" substitutionGroup="dc:relation">
   <xs:complexType>
      <xs:complexContent>
         <xs:extension base="dc:SimpleLiteral">
            <xs:attributeGroup ref="xlink:simpleLink"/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:element>
<xs:element name="relations">
   <xs:complexType>
      <xs:sequence>
         <xs:element ref="relation" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
<xs:element name="Resources">
   <xs:complexType>
      <xs:sequence>
         <xs:element ref="ddms:Resource" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

The **Resources** element contains zero or more **ddms:Resource** elements used to describe zero or more resources in the containing XML schema document. The **relations** element contains zero or more **relation** elements used to declare the relationships among the resources according to the UML diagram (figure X). Resources were referenced using their identifier encoded as a UUID. The following XML fragment is an example of the **relations** element that might appear in an XML schema appinfo annotation:

```
<relations xmlns="http://www.opengis.net/ows3/geodss/schemametadata">
   <relation
    xlink:href="urn:uuid:96284A9E-8074-44E2-AF99-012771B4963F">derivedFrom</relation>
   <relation
    xlink:href="urn:uuid:c4a0f991-fe30-4e5f-9da3-f7b986885aaf">dependsOn</relation>
   <relation xlink:href="out/MSD3.xsd">resource</relation>
</relations>
```

In this example the resource, the schema document MSD3.xsd, depends on the resource with id *urn:uuid:c4a0f991-fe30-4e5f-9da3-f7b986885aaf* and is derived from the resource with id *urn:uuid:96284A9E-8074-44E2-AF99-012771B4963F*.

Table 13 maps the elements of the DDMS to the ebRIM:

Table 13 – Mapping DDMS Elements to ebRIM

DDMS element	ebRIM element
identifier identifier/@qualifier identifier/@value	/ExternalIdentifier[@registryObject="*id of DDMS resource*"] /ExternalIdentifier/@identificationScheme /ExternalIdentifier/@value
title	/ExtrinsicObject[@objectType="DDMS"]/Slot[@name="title"]/ ValueList/Value[0] /ExtrinsicObject[@objectType="DDMS"]/Slot[@name="title"]/ ValueList/Value[1-N]
subtitle	/ExtrinsicObject[@objectType="DDMS"]/Slot[@name="subtitle"]/ ValueList/Value[0] /ExtrinsicObject[@objectType="DDMS"]/Slot[@name="subtitle"]/ ValueList/Value[1-N]

description	/ExtrinsicObject[@objectType="DDMS"]/Description/LocalizedString@value /ExtrinsicObject[@objectType="DDMS"]/Slot[@name="*name of ICSM marking*"]/…
language	/ExtrinsicObject[@objectType="DDMS"]/Slot[@name="language"]/ ValueList/[Value="*qualifier:value*"
dates	/ExtrinsicObject[@objectType="DDMS"]/Slot[@name="created"]/… /ExtrinsicObject[@objectType="DDMS"]/Slot[@name="posted"]/… /ExtrinsicObject[@objectType="DDMS"]/Slot[@name="validTil"]/… /ExtrinsicObject[@objectType="DDMS"]/Slot[@name="infoCutOff"]/…
Source	/ExternalLink/Slot[@name="qualifier"]/… /ExternalLink/Slot[@name="value"]/… /ExternalLink/Slot[@name="schemaQualifier"]/… /ExternalLink/@externalURI /Association[@sourceObject="*id of DDMS resource*"] /Association[@assocationType="source"] /Association[@targetObject="*id of ExternalLink*"]
Type	/ExtrinsicObject[@objectType="DDMS"]/Slot[@name="type"]/ ValueList/[Value="*qualifier:value*"]
Creator publisher pointOfContact ontributor	/Resource/…/Organization -> /Organization /Resource/…/Person -> /User /Resource/…/Service -> /Service /Association[@sourceObject="*id of DDMS resource*"] /Association[@associationType="creator | publisher | pointOfContact | contributor"] /Association[@targetObject="*id of Org/User/Service*"] /Association/Slot[@"*name of ICISM marking*"]/ValueList/Value
Rights	/ExtrinsicObject[@objectType="DDMS"]/Slot[@name="privacyAct"]/… /ExtrinsicObject[@objectType="DDMS"/Slot[@name="intellectualProperty"]… /ExtrinsicObject[@objectType="DDMS"]/Slot[@name="copyright"]/…
Format	/ExtrinsicObject[@objectType="DDMS"]/@mimeType /ExtrinsicObject[@objectType="DDMS"]/Slot[@name="extent]/ ValueList/[Value="*qualifier:value*"] /ExtrinsicObject[@objectType="DDMS"]/Slot[@name="medium"/…
subjectCoverage/ Subject/keyword subjectCoverage/ Subject/category@qualifier subjectCoverage/ Subject/category@label	 /ExtrinsicObject[@objectType="DDMS"]/Slot[@name="Keyword]/… /Classification@classificationScheme /Classification@classificationNode
TemporalCoverage/ TimePeriod	/ExtrinsicObject[@objectType="DDMS"]/Slot[@name= "temporalCoverage"]/ValueList/[Value="ISO8601 Period"]
geospatialCoverage/ Place|Facility geospatialCoverage/ Place | Facility/georef	/Organization /Association[@sourceObject="*id of resource*"] /Association[@assocationType="geospatialCoverage] /Assocation[@targetObject="*id of Organization*"] /ExtrinsicObject[objectType="DDMS"]/Slot[name="Footprint"]/…
security	/ExtrinsicObject[@objectType="Security]/Slot[@name="classification"]/… /ExtrinsicObject[@objectType="Security]/Slot[@name="ownerProducer"]/… /ExtrinsicObject[@objectType="Security]/Slot[@name="SCIcontrols"]/… /ExtrinsicObject[@objectType="Security]/Slot[@name="SARIdentifier"]/… /ExtrinsicObject[@objectType="Security]/Slot[@name="disseminationControls"]/… /ExtrinsicObject[@objectType="Security]/Slot[@name="FGIsourceOpen"]/…

	/ExtrinsicObject[@objectType="Security]/Slot[@name="FGIsourceProtected"]/… /ExtrinsicObject[@objectType="Security]/Slot[@name="releasableTo"]/… /ExtrinsicObject[@objectType="Security]/Slot[@name="nonICmarkings"]/… /ExtrinsicObject[@objectType="Security]/Slot[@name="classifiedBy"]/… /ExtrinsicObject[@objectType="Security]/Slot[@name="classificationReason"]/… /ExtrinsicObject[@objectType="Security]/Slot[@name="derivedFrom"]/… /ExtrinsicObject[@objectType="Security]/Slot[@name="declassDate"]/… /ExtrinsicObject[@objectType="Security]/Slot[@name="declassEvent"]/… /ExtrinsicObject[@objectType="Security]/Slot[@name="declassException"]/… /ExtrinsicObject[@objectType="Security]/Slot[@name="typeOfExemptedSource"]/… /ExtrinsicObject[@objectType="Security]/Slot[@name="dateOfExemptedSource"]/… /ExtrinsicObject[@objectType="Security]/Slot[@name="declassManualReview"]/… /Association[@sourceObject="*id of resource*"] /Association[@assocationType="securityMarkings"] /Assocation[@targetObject="*id of Security resourcet*"]

11.1 Information security markings

The DDMS uses the *IC-ISM-v2.xsd* schema to security tag information in the metadata document. Security markings exist in two places in a DDMS document. First, they may be placed on individual elements to tag the content of the elements (e.g. the description element) or they may be placed on the **security** element that applies to the document as a whole.

In the case of the **security** element, an extrinsic object of type *Security* was created containing slots whose names correspond to the IC-ISM names. This object was then associated with the source object using the *securityMarkings* association type (see Table 13).

Otherwise, the ICSM values where added to the list of values for a slot whose name corresponds to the name of the element to which the security marking apply and whose values have the format "*ISCM name*=value". For example, consider the DDMS element named *title*. This element is mapped to a slot in the ebRIM named *title*. Any ICSM values would be mapped to this slot as follows:

```
/ExtrinsicObject/Slot[@name="title"]/ValueList/[Value[0]="title"]
/ExtrinsicObject/Slot[@name="title"]/ValueList/[Value[1]="classification=value"]
/ExtrinsicObject/Slot[@name="title"]/ValueList/[Value[2]="ownerProducer=value"]
/ExtrinsicObject/Slot[@name="title"]/ValueList/[Value[3]="SCIcontrols=value"]
/ExtrinsicObject/Slot[@name="title"]/ValueList/[Value[4]="SARIdentifier=value"]
/ExtrinsicObject/Slot[@name="title"]/ValueList/[Value[5]="disseminationControls=value"]
/ExtrinsicObject/Slot[@name="title"]/ValueList/[Value[6]="FGIsourceOpen=value"]
/ExtrinsicObject/Slot[@name="title"]/ValueList/[Value[7]="FGIsourceProtected=value"]
/ExtrinsicObject/Slot[@name="title"]/ValueList/[Value[8]="releasableTo=value"]
/ExtrinsicObject/Slot[@name="title"]/ValueList/[Value[9]="nonICmarkings=value"]
/ExtrinsicObject/Slot[@name="title"]/ValueList/[Value[10]="classifiedBy=value"]
/ExtrinsicObject/Slot[@name="title"]/ValueList/[Value[11]="classificationReason=value"]
/ExtrinsicObject/Slot[@name="title"]/ValueList/[Value[12]="derivedFrom=value"]
/ExtrinsicObject/Slot[@name="title"]/ValueList/[Value[13]="declassDate=value"]
/ExtrinsicObject/Slot[@name="title"]/ValueList/[Value[14]="declassEvent=value"]
/ExtrinsicObject/Slot[@name="title"]/ValueList/[Value[15]="declassException=value"]
/ExtrinsicObject/Slot[@name="title"]/ValueList/[Value[16]="typeOfExemptedSource=value"]
/ExtrinsicObject/Slot[@name="title"]/ValueList/[Value[17]="dateOfExemptedSource=value"]
/ExtrinsicObject/Slot[@name="title"]/ValueList/[Value[18]="declassManualReview=value"]
```

12 Symbology encoding documents

Table 14 defines the mapping between the elements in a symbology encoding document to the elements of the ebRIM.

Table 14 – Mapping SE Elements to ebRIM

Symbology Encoding element	EbRIM element
/SymbologyEncoding/FeatureTypeStyle/Name	/ExtrinsicObject[@objectType="Symbology Encoding Document"]/Name/LocalizedString/Value
/SymbologyEncoding/FeatureTypeStyle/Title	/ExtrinsicObject[@objectType="Symbology Encoding Document"]/Slot[@name="Title"]/ValueList/Value
/SymbologyEncoding/FeatureTypeStyle/Abstract	/ExtrinsicObject[@objectType="Symbology Encoding Document"]/Description/LocalizedString/Value

The relationship between a symbology encoding document and the feature that it symbolizes is encoded by creating an **Association** record. The source id of the association is the id of the catalog record for the symbology encoding document. The target id of the association is set to the id of the catalog record for the feature type that is being symbolized. The association type is set to "*Symbolizes*".

13 SWE metadata documents

T.B.D.

14 Service Binding

14.1 Introduction

So far, this documents has been concerned with mapping various types of metadata into the catalog's information model, ebRIM. This sections, however, deals with using that registered information.

A common usage pattern for the catalog is to search for data resources and then, having found one or more data resources of interest, locate and invoke a service that offers that data. This process is called *binding* to the service. This section discusses how the catalog can be queried to locate services that offer data of interest and then invoke those services.

The basic mechanism that supports binding is the *association*. That is that there must exist a means of describing an association between one catalogued object and another catalogued object. Once the association is established between a data item and a service that offers that data item, find one participant in the association means that the other participant can also be located. So finding the data mans that services offering the data can be found as well as invocation instructions for those services (e.g. WSDL documents, specification documents).

14.2 Service binding with the common profile

Resources are related in the common profile using the **dc:Related** element which can be repeated as many times are required to enumerate all the associations that a particular object has. Having found a data item in the catalogue, the process of locating a service that offers that data items involves search for all related object of type *service*. The following XML fragment shows what such a query might look like:

```
<GetRecords>
   <Query typeName="csw:Record=A csw:Record=B>
      <ElementName>/A</ElementName>
      <Constraint>
         <ogc:Filter>
            <And>
               <ogc:PropertyIsEqualTo>
                  <ogc:PropertyName>/A@id</ogc:PropertyName>
                  <ogc:PropertyName>urn:uuid:…</ogc:PropertyName>
               </ogc:PropertyIsEqual>
               <ogc:PropertyIsEqualTo>
                  <ogc:PropertyName>/B@id</ogc:PropertyName>
                  <ogc:PropertyName>/A/dc:related</ogc:PropertyName>
               </ogc:PropertyIsEqual>
               <ogc:PropertyIsEqualTo>
                  <ogc:PropertyName>/B/dc:type</ogc:PropertyName>
                  <ogc:Literal>Service</ogc:Literal>
               </ogc:PropertyIsEqual
            </And>
         </ogc:Filter>
      </Constraint>
   </Query>
</GetRecords>
```

This query finds all catalog records of type *Service* related to a source record identified by the string *urn:uuid:….* Additional constraints can be added to this query to narrow the list of candidate service records even further. Once a candidate service has been identified, binding information must be obtained for the service.

For the common profile, binding information can take a number of forms. First, the service may include additional **dc:identifier** elements that contain a URI pointing to an access point for the service. Additionally, or instead of, the service record may contain a reference to a WSDL document using the **dc:relation** element. During the course of OWS3 both methods were used and neither proved a completely satisfactory method of binding to a service. The best approach was to include an additional dc:identifier element whose value was a URI that resolved to the capabilities document of the service.

14.3 Service binding with the ebRIM profile

Once a data resource has been identified in the catalog, the **Association** entity can then be used to locate services that offer that resource. The following XML fragment shows a query finding all services that offer a data resource:

```
<GetRecords>
   <Query typeNames="Service Association">
      <ElementName>/Service</ElementName>
      <Constraint>
         <ogc:Filter>
            <ogc:And>
```

```
               <ogc:PropertyIsEqualTo>
                  <ogc:PropertyName>/Association@targetObject</ogc:PropertyName>
                  <ogc:Literal>urn:uuid:…</ogc:Literal>
               </ogc:PropertyIsEqualTo>
               <ogc:PropertyIsEqualTo>
                  <ogc:PropertyName>/Association@associationType</ogc:PropertyName>
                  <ogc:Literal>Offers</ogc:Literal>
               </ogc:PropertyIsEqualTo>
               <PropertyIdEqualTo>
                  <ogc:PropertyName>/Service@id</ogc:PropertyName>
                  <ogc:PropertyName>/Association@sourceObject</ogc:PropertyName>
               </ogc:PropertyIsEqualTo>
            </ogc:And>
         </ogc:Filter>
      </Constraint>
   </Query>
</GetRecords>
```

Each service records returned by this query will include **rim:ServiceBinding** elements that include access point information for the various operations and HTTP bindings that the service offers. Sections 8.3.1, 9.4 and 10.3 describe how this binding information is mapped into the ebRIM for WMS, WFS and WCS services.

14.4 Service binding with the ISO profile

T.B.D.

15 Common profile

15.1 Introduction

The purpose of this section is to describe how clients that support implement the CSW 2.0, HTTP protocol binding may interoperably interact with ebRIM and ISO profile catalogs.

Although the ebRIM and ISO based catalog profiles use different information models (ebRIM or ISO19119/ISO19115), it is possible for a catalog client to query each catalog and obtain useful results using a common record schema. This is achieved by using the common profile defined in the CSW 2.0 specification. The salient feature of the common profile is the information model which declares the **csw:Record** element. The **csw:Record** element is the root element of the common profile's imformation model and is a container for a subset of Dublin Core metadata elements.

According to the CSW 2.0 specification, all compliant catalogs must support a view of their information model that maps to the **record.xsd** schema . Thus, a client should be able to query any CSW 2.0 catalog, regardless of the underlying information model, using the elements defined in the **record.xsd** schema. The **record.xsd** schema can be found at:

http://schemas.opengis.net/csw/2.0.0/record.xsd

The schema defines brief, summary and full representations. The corresponding root elements are ... **csw:BriefRecord**, **csw:SummaryRecord** and **csw:Record** for the full set of elements.

15.2 Core queryables

The OGC core queryables are a set of abstract query points based on the Dublin Core set of metadata elements. They are referred to as abstract because these are mapped to the internal information model of the catalog. It is this mapping that allows cross-catalog queries to be performed since the details of the internal information model are opaque to clients.

The following table lists the set of core queryable elements with a short narrative outlining of what each element describes. In addition to the elements described here, there is also a set of alternate terms defined in the http://schemas.opengis.net/csw/2.0.0/rec-dcterms.xsd schema.

Table 15 – OGC Common Profile Elements

OGC Core Element	Description
Title	A name given to the resource. Typically, Title will be a name by which the resource is formally known.
Creator	An entity primarily responsible for making the content of the resource. Examples of Creator include a person, an organization, or a service. Typically, the name of a Creator should be used to indicate the entity.
Subject	A topic of the content of the resource. Typically, Subject will be expressed as keywords, key phrases, or classification codes that describe a topic of the resource. Recommended best practice is to select a value from a controlled vocabulary or formal classification scheme.
Description	An account of the content of the resource. Examples of Description include, but are not limited to, an abstract, table of contents, reference to a graphical representation of content, or free-text account of the content.
Publisher	An entity responsible for making the resource available. Examples of Publisher include a person, an organization, or a service. Typically, the name of a Publisher should be used to indicate the entity.
contributor	An entity responsible for making contributions to the content of the resource. Examples of Contributor include a person, an organization, or a service. Typically, the name of a Contributor should be used to indicate the entity.
date	A date of an event in the lifecycle of the resource. Typically, Date will be associated with the creation or availability of the resource. Recommended best practice for encoding the date value is defined in a profile of ISO 8601 and includes (among others) dates of the form YYYY-MM-DD.
type	The nature or genre of the content of the resource. Type includes terms describing general categories, functions, genres, or aggregation levels for content. Recommended best practice is to select a value from a controlled vocabulary (for example, the DCMI Type Vocabulary). To describe the physical or digital manifestation of the resource, use the Format element.
format	The physical or digital manifestation of the resource. Typically, Format will include the media-type or dimensions of the resource. Format may be used to identify the software, hardware, or other equipment needed to display or operate the resource. Examples of dimensions include size and duration. Recommended best practice is to select a value from a controlled vocabulary (for example, the list of Internet Media Types defining computer media formats).
identifier	An unambiguous reference to the resource within a given context. Recommended best practice is to identify the resource by means of a string or number conforming to a formal identification system. Formal identification systems include but are not limited to the Uniform Resource Identifier (URI) (including the Uniform Resource Locator (URL)), the Digital Object Identifier (DOI), and the International Standard Book Number (ISBN).
source	A Reference to a resource from which the present resource is derived. The present resource may be derived from the Source resource in whole or in part. Recommended best practice is to

	identify the referenced resource by means of a string or number conforming to a formal identification system.
language	A language of the intellectual content of the resource. Recommended best practice is to use RFC 3066, which, in conjunction with ISO 639, defines two- and three-letter primary language tags with optional subtags. Examples include “en” or “eng” for English, “akk" for Akkadian, and “en-GB” for English used in the United Kingdom.
relation	A reference to a related resource. Recommended best practice is to identify the referenced resource by means of a string or number conforming to a formal identification system.
coverage	The extent or scope of the content of the resource. Typically, Coverage will include spatial location (a place name or geographic coordinates), temporal period (a period label, date, or date range), or jurisdiction (such as a named administrative entity). Recommended best practice is to select a value from a controlled vocabulary (for example, the Thesaurus of Geographic Names [TGN]) and to use, where appropriate, named places or time periods in preference to numeric identifiers such as sets of coordinates or date ranges.
rights	Information about rights held in and over the resource. Typically, Rights will contain a rights management statement for the resource, or reference a service providing such information. Rights information often encompasses Intellectual Property Rights (IPR), Copyright, and various Property Rights. If the Rights element is absent, no assumptions may be made about any rights held in or over the resource.
BoundingBox	Describing the spatial extent of the resource.

The **record.xsd** schema defines three elements sets:

Table 16 – Element Set Names

Element Set Name	OGC core elements
Brief	Identifier, type
Summary	identifier, type, title, subject, format, relation, date, description, BoundingBox
Full	the full set of element

The following XML fragments, taken from the schema tailoring effort in OWS 3, are examples of each element set.

Brief Record Example:

```
<csw:BriefRecord xmlns:dct="http://www.purl.org/dc/terms/"
  xmlns:dc="http://www.purl.org/dc/elements/1.1/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.purl.org/dc/terms/
                      http://schemas.opengis.net/csw/2.0.0/rec-dcterms.xsd
                      http://www.purl.org/dc/elements/1.1/
                      http://schemas.opengis.net/csw/2.0.0/rec-dcmes.xsd">
  <dc:identifier>f81d4fae-7dec-11d0-a765-00a0c91e6bf6</dc:identifier>
  <dc:type>UML Model</dc:type>
</csw:BriefRecord>
```

Summary Record Example:

```
 <csw:SummaryRecord xmlns:dct="http://www.purl.org/dc/terms/"
  xmlns:dc="http://www.purl.org/dc/elements/1.1/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.purl.org/dc/terms/
                      http://schemas.opengis.net/csw/2.0.0/rec-dcterms.xsd
```

```
                        http://www.purl.org/dc/elements/1.1/
                        http://schemas.opengis.net/csw/2.0.0/rec-dcmes.xsd">
      <dc:identifier>f81d4fae-7dec-11d0-a765-00a0c91e6bf6</dc:identifier>
      <dc:type>UML Model</dc:type>
      <dc:title>OWS Test</dc:title>
      <dc:subject>MSD1<dc:subject>
      <dc:format>text/xml; subtype="xmi/1.0/iso19136"</dc:format>
      <dc:relation xlink:href="urn:uuid:f264-77d2-09ce-aa39-e246">dependsOn</dc:relation>
      <dct:modified>2005-07-18</dct:modified>
      <dct:abstract>MSD Level 1 application schema. Automatically created from the NGA Access
Databases (FACC/DFDD, NGA Feature Catalogue and Mission Specific Data).</dct:abstract>
    </csw:SummaryRecord>
```

Full Record Example:

```
    <csw:Record xmlns:dct="http://www.purl.org/dc/terms/"
      xmlns:dc="http://www.purl.org/dc/elements/1.1/"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.purl.org/dc/terms/
                        http://schemas.opengis.net/csw/2.0.0/rec-dcterms.xsd
                        http://www.purl.org/dc/elements/1.1/
                        http://schemas.opengis.net/csw/2.0.0/rec-dcmes.xsd">
      <dc:identifier>f81d4fae-7dec-11d0-a765-00a0c91e6bf6</dc:identifier>
      <dc:type>UML Model</dc:type>
      <dc:title>OWS Test</dc:title>
      <dc:subject>MSD1<dc:subject>
      <dc:format>text/xml; subtype="xmi/1.0/iso19136"</dc:format>
      <dc:relation xlink:href="urn:uuid:f264-77d2-09ce-aa39-e246">dependsOn</dc:relation>
      <dct:modified>2005-07-18</dct:modified>
      <dct:abstract>MSD Level 1 application schema. Automatically created from the NGA Access
Databases (FACC/DFDD, NGA Feature Catalogue and Mission Specific Data).</dct:abstract>
      <dct:created>2005-04-01</dct:created>
      <dc:creator>NGA703-814-4580NCGIS-mail@nga.mil</dc:creator>
      <dc:language>en</dc:language>
      <dc:publisher>NGA</dc:publisher>
    </csw:Record>
```

15.3 Catalog operations

15.3.1 Introduction

This section discusses the operations defined for the HTTP protocol binding with respect to the OGC core elements. In each case, issue regarding the processing of OGC core elements are highlighted.

15.3.2 GetCapabilities operation

In order for the client to know which core queryable elements sets are available for query, a catalog should advertise this information in the get capabilities response.

The approach taken is to rely on the OGC common capabilities schema and simply list the elements that are query targets in the parameter lists of operations that apply (e.g. **GetRecords**).

The following XML fragment from a capabilities document shows an **Operation** element describing the **csw:GetRecords** operation. The **Parameter** named *typeName* is used to

advertise the queryable elements including the **csw:BriefRecord**, **csw:SummaryRecord** and full **csw:Record** elements of the OGC core queryables.

```
<ows:Operation name="GetRecords">
 <ows:DCP>
    <ows:HTTP>
       <ows:Get xlink:type="simple" xlink:href="http://www...com/cwwrs/cwwrs.cgi?"/>
       <ows:Post xlink:type="simple" xlink:href="http://www...com/cwwrs/cwwrs.cgi"/>
    </ows:HTTP>
 </ows:DCP>
 <ows:Parameter name="typeNames">
    <ows:Value>csw:BriefRecord<ows:Value>
    <ows:Value>csw:SummaryRecord</ows:Value>
    <ows:Value>csw:Record</ows:Value>
    <ows:Value>ebrim:Association</ows:Value>
    <ows:Value>ebrim:AuditableEvent</ows:Value>
    <ows:Value>ebrim:AffectedObject</ows:Value>
    <ows:Value>ebrim:Classification</ows:Value>
    <ows:Value>ebrim:ClassificationNode</ows:Value>
    <ows:Value>ebrim:ClassificationScheme</ows:Value>
    <ows:Value>ebrim:ExternalIdentifier</ows:Value>
    <ows:Value>ebrim:ExternalLink</ows:Value>
    <ows:Value>ebrim:ExtrinsicObject</ows:Value>
    <ows:Value>ebrim:Federation</ows:Value>
    <ows:Value>ebrim:Organization</ows:Value>
    <ows:Value>ebrim:Registry</ows:Value>
    <ows:Value>ebrim:Service</ows:Value>
    <ows:Value>ebrim:ServiceBinding</ows:Value>
    <ows:Value>ebrim:SpecificationLink</ows:Value>
    <ows:Value>ebrim:Subscription</ows:Value>
    <ows:Value>ebrim:AdhocQuery</ows:Value>
    <ows:Value>ebrim:User</ows:Value>
 </ows:Parameter>
 .
 .
 .
</ows:Operation>
```

What this XML fragment is saying is that this catalog supports the **GetRecords** operation (at the specified GET and POST URLs) and the value of the *typeNames* parameter is limited to the set of elements listed in the fragment.

Since the catalog is ebRIM based, most of the elements are ebRIM elements. However, the first elements in the list are **csw:BriefRecord**, **csw:SummaryRecord** and **csw:Record** indicating the OGC core queryable elements sets that the catalogue supports.

15.3.3 DescribeRecord operation

A catalog must be able to describe the information model that it uses. This description is usually expressed as an XML-Schema document in response to a **DescribeRecord** request. In the case of the OGC core queryables, the catalog simply needs to respond with the http://schemas.opengis.net/csw/2.0.0/record.xsd schema.

15.3.4 GetRecords operation

Querying the OGC core queryable properties should be straight forward since the schema is essentially a flat list of properties. Here is an example query against the OGC core queryable properties:

```
<GetRecords
   xmlns="http://www.opengis.net/csw"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:dc="http://www.purl.org/dc/elements/1.1/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation=
      "http://www.opengis.net/csw
       http://schemas.opengis.net/csw/2.0.0/CSW-discovery.xsd
       http://www.opengis.net/ogc
       http://schemas.opengis.net/filter/1.0.0/filter.xsd
       http://www.purl.org/dc/elements/1.1/
       http://schemas.opengis.net/csw/2.0.0/rec-dcmes.xsd"
   version="2.0.0"
   service="CSW"
   outputFormat="text/xml"
   outputSchema="http://schemas.opengis.net/csw/2.0.0/record.xsd">
   <Query typeNames="csw:Record">
      <ElementSetName>full</ElementSetName>
      <Constraint>
         <ogc:Filter>
            <ogc:PropertyIsEqualTo>
               <ogc:PropertyName>/csw:Record/dc:publisher</ogc:PropertyName>
               <ogc:Literal>NGA</ogc:Literal>
            </ogc:PropertyIsEqualTo>
         </ogc:Filter>
      </Constraint>
   </Query>
</GetRecords>
```

A sample response to this query might be:

```
<csw:GetRecordsResponse
     xmlns:csw="http://www.opengis.net/cat/csw"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://www.opengis.net/cat/csw
                    http://schemas.opengis.net/csw/2.0.0/CSW-discovery.xsd"
     version="2.0.0">
   <csw:SearchStatus status="complete" timestamp="2005-07-18T10:13:27"/>
   <csw:SearchResults
        recordSchema="http://schemas.opengis.net/csw/2.0.0/record.xsd"
        numberOfRecordsMatched="1" numberOfRecordsReturned="1">
       <csw:Record xmlns:dct="http://www.purl.org/dc/terms/"
          xmlns:dc="http://www.purl.org/dc/elements/1.1/"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.purl.org/dc/terms/
                              http://schemas.opengis.net/csw/2.0.0/rec-dcterms.xsd
                              http://www.purl.org/dc/elements/1.1/
                              http://schemas.opengis.net/csw/2.0.0/rec-dcmes.xsd">
          <dc:identifier>f81d4fae-7dec-11d0-a765-00a0c91e6bf6</dc:identifier>
          <dc:type>UML Model</dc:type>
          <dc:title>OWS Test</dc:title>
          <dc:subject>MSD1<dc:subject>
          <dc:format>text/xml; subtype="xmi/1.0/iso19136"</dc:format>
          <dc:relation xlink:href="urn:uuid:f264-77d2-09ce-aa39-
e246">dependsOn</dc:relation>
          <dct:modified>2005-07-18</dct:modified>
          <dct:abstract>MSD Level 1 application schema. Automatically created from the NGA
Access Databases (FACC/DFDD, NGA Feature Catalogue and Mission Specific Data).</dct:abstract>
          <dct:created>2005-04-01</dct:created>
          <dc:creator>NGA703-814-4580NCGIS-mail@nga.mil</dc:creator>
          <dc:language>en</dc:language>
          <dc:publisher>NGA</dc:publisher>
       </csw:Record>
   </csw:SearchResults>
</csw:GetRecordsResponse>
```

The content of the **csw:SearchResults** element depends of the value of the **csw:ElementSetName** element in the query. In this case, the value full causes all available OGC core elements to be presented.

Since all catalogs that comply with the CSW 2.0 specification must support a view of the catalog based on the **csw:Record** schema, this example query should run on any CSW 2.0 catalog regardless of the particular information model being used. So, this query should run unchanged on ebRIM based and ISO19115/19119 based catalogs.

15.3.4.1 Equivalence of common profile and ebRIM

The next two examples are meant to show the same query ... one encoded using the **csw:Record** schema and the other encoded using the ebRIM schema. This query is looking for WMS layers whose description contain the keyword "school" and which lie within a particular area of interest. The first query is against the **csw:Record** schema. Then second, equivalent query, is against the ebRIM schema.

Of course, equivalence can only be established as long as the properties being queried are mapable between ebRIM and CSW.

ebRIM example:

```
<GetRecords
   xmlns="http://www.opengis.net/cat/csw"
   xmlns:dc="http://www.purl.org/dc/elements/1.1/"
   xmlns:csw="http://www.opengis.net/cat/csw"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:ows="http://www.opengeospatial.net/ows"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation=
      "http://www.opengis.net/cat/csw
       http://schemas.opengis.net/csw/2.0.0/CSW-discovery.xsd
       http://www.opengis.net/ogc
       http://schemas.opengis.net/filter/1.0.0/filter.xsd
       http://www.opengeospatial.net/ows
       http://schemas.opengis/net/ows/1.0.0/owsAll.xsd
       http://www.purl.org/dc/elements/1.1/
       http://schemas.opengis.net/csw/2.0.0/rec-dcmes.xsd"
   service="CSW"
   version="2.0.0"
   outputFormat="text/xml"
   outputSchema="http://schemas.opengis.net/csw/2.0.0/record.xsd">
   <Query typeNames="csw:Record">
      <Constraint version="1.0.0">
         <ogc:Filter>
            <ogc:And>
               <ogc:Or>
                 <ogc:PropertyIsLike escape="\" singleChar="_" wildCard="%">
                    <ogc:PropertyName>/csw:Record/dc:name</ogc:PropertyName>
                    <ogc:Literal>%school%</ogc:Literal>
                 </ogc:PropertyIsLike>
                 <ogc:PropertyIsLike escape="\" singleChar="_" wildCard="%">
                    <ogc:PropertyName>/csw:Record/dc:description</ogc:PropertyName>
                    <ogc:Literal>%school%</ogc:Literal>
                 </ogc:PropertyIsLike>
              </ogc:Or>
              <ogc:Or>
                 <ogc:PropertyIsEqualTo>
                    <ogc:PropertyName>/Record/dc:type</ogc:PropertyName>
```

```
                  <ogc:Literal>FeatureType</ogc:Literal>
               </ogc:PropertyIsEqualTo>
               <ogc:PropertyIsEqualTo>
                  <ogc:PropertyName>/Record/dc:type</ogc:PropertyName>
                  <ogc:Literal>WMS_Layer</ogc:Literal>
               </ogc:PropertyIsEqualTo>
            </ogc:Or>
            <ogc:Contains>
               <ogc:PropertyName>/Record/ows:BoundingBox</ogc:PropertyName>
               <gml:Box srsName="EPSG:4326">
                  <gml:coordinates>-80,30 -70,40</gml:coordinates>
               </gml:Box>
            </ogc:Contains>
         </ogc:And>
      </ogc:Filter>
    </Constraint>
  </Query>
</GetRecords>
```

OGC core queryables example:

```
<GetRecords
   xmlns="http://www.opengis.net/cat/csw"
   xmlns:ebxml="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:ows="http://www.opengeospatial.net/ows"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation=
      "http://www.opengis.net/cat/csw
       http://schemas.opengis.net/csw/2.0.0/CSW-discovery.xsd
       http://www.opengis.net/ogc
       http://schemas.opengis.net/filter/1.0.0/filter.xsd
       http://www.opengeospatial.net/ows
       http://schemas.opengis/net/ows/1.0.0/owsAll.xsd
       urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5
       http://www.pvretano.com/schemas/ebrim/2.5/rim.xsd"
   service="CSW"
   version="2.0.0"
   outputFormat="text/xml"
   outputSchema="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5">
  <Query typeNames="ExtrinsicObject">
    <ElementName>/ExtrinsicObject</ElementName>
    <Constraint version="1.0.0">
      <ogc:Filter">
        <ogc:And>
          <ogc:Or>
            <ogc:PropertyIsLike escape="\" singleChar="_" wildCard="%">

<ogc:PropertyName>/ExtrinsicObject/Name/LocalizedString/@value</ogc:PropertyName>
              <ogc:Literal>%school%</ogc:Literal>
            </ogc:PropertyIsLike>
            <ogc:PropertyIsLike escape="\" singleChar="_" wildCard="%">

<ogc:PropertyName>/ExtrinsicObject/Description/LocalizedString/@value</ogc:PropertyName>
              <ogc:Literal>%school%</ogc:Literal>
            </ogc:PropertyIsLike>
          </ogc:Or>
          <ogc:Or>
            <ogc:PropertyIsEqualTo>
              <ogc:PropertyName>/ExtrinsicObject/@objectType</ogc:PropertyName>
              <ogc:Literal>FeatureType</ogc:Literal>
            </ogc:PropertyIsEqualTo>
            <ogc:PropertyIsEqualTo>
              <ogc:PropertyName>/ExtrinsicObject/@objectType</ogc:PropertyName>
              <ogc:Literal>WMS_Layer</ogc:Literal>
            </ogc:PropertyIsEqualTo>
          </ogc:Or>
          <ogc:Contains>
```

```
<ogc:PropertyName>/ExtrinsicObject/Slot[@name="FootPrint"]/ValueList/value[1]</ogc:PropertyNa
me>
               <gml:Box srsName="EPSG:4326">
                 <gml:coordinates>-80,30 -70,40</gml:coordinates>
               </gml:Box>
             </ogc:Contains>
           </ogc:And>
         </ogc:Filter>
       </Constraint>
     </Query>
   </GetRecords>
```

15.3.5 Transaction operation

Since a catalog may be based on any information model, it may or may not support transactions based on the **csw:Record** schema. Typically it is expected that catalogs would not support the Transaction operation of the csw:Record schema except in the case where the only information model that the catalog support is the csw:Record schema.

15.3.6 New operations

As described in clause 6.2.1 may cache the resource being harvested in a local repository. This means that a new catalog operation needs to be added to allow this content to be retrieved based on the record identifier.

The ebRIM catalog profile (05-025r3) defines an operation called **GetRepositoryItem**. The purpose of this operation is to allow items stored in a catalog's repository to be retrieved based on a record id. This operation is distinct from the **GetRecordById** function since the **GetRecordById** operation fetches a record in the schema of the information model of the catalog. The **GetRepositoryItem** operation fetches the object, stored in the catalog's repository, the caused the record to be created in the first place. Referring to the workflow described in clause 6.2.1, the **GetRepositoryItem** operation fetches the optionally cached source metadata document (step 1a).

Continuing the example from clause 11.2, the response to the request:

```
<GetRepositoryItem>
   <Id>urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6</Id>
</GetRepositoryItem>
```

would be the XMI document that the catalog record with id *urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6* is describing.

15.3.6.1 csw:Record attribute "contents"

Another, perhaps complimentary approach to catalog interoperability would be to extend the **csw:Record** schema to include a **contents** attribute, defined as a link which always points at the catalogued content, whether it is the record type itself stored directly in the catalog, the item in a local ebRIM repository pointed to by the **ExtrinsicObject**, or a remote resource altogether. This would resolve differences between catalog profiles in the number of levels of indirection as well as in the particular repository interface that has been implemented.

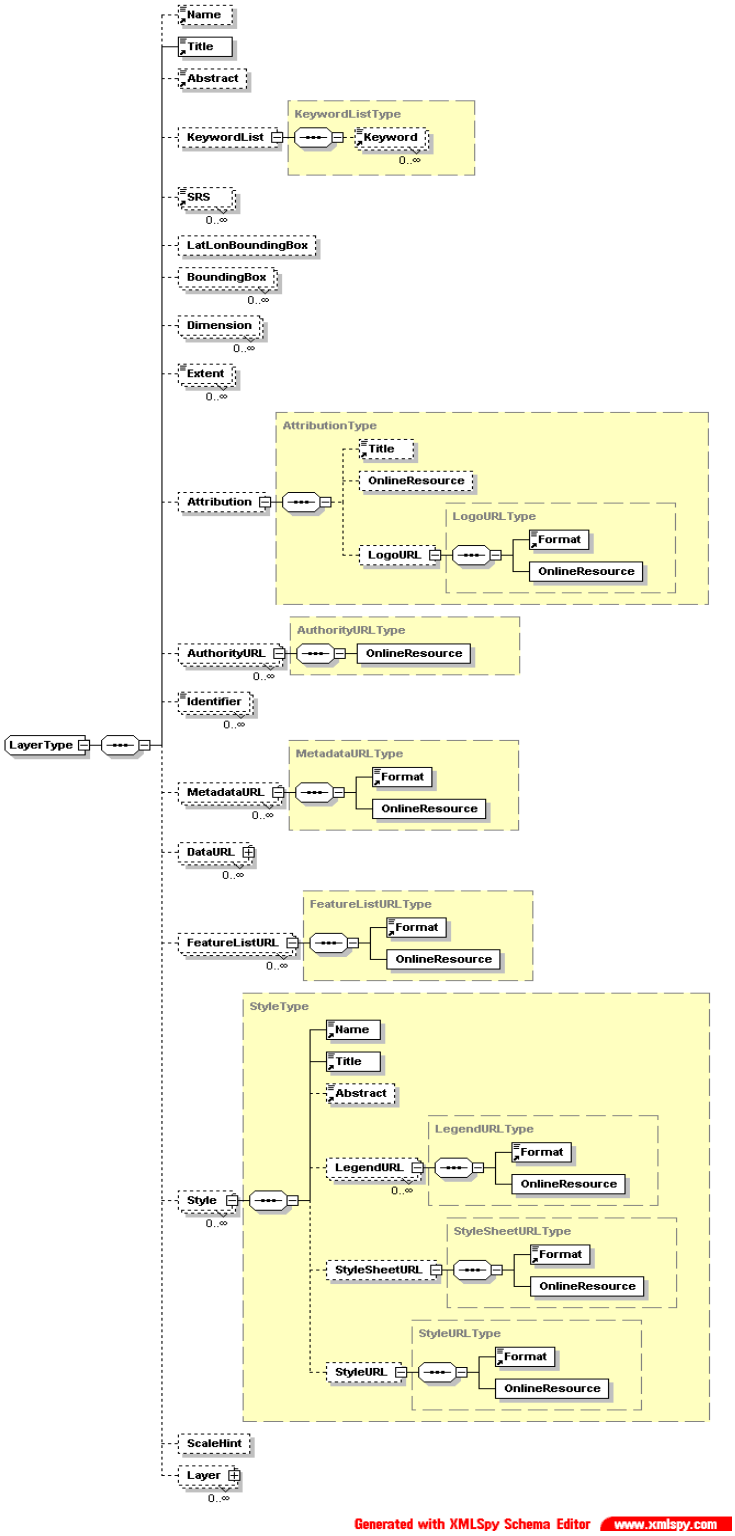
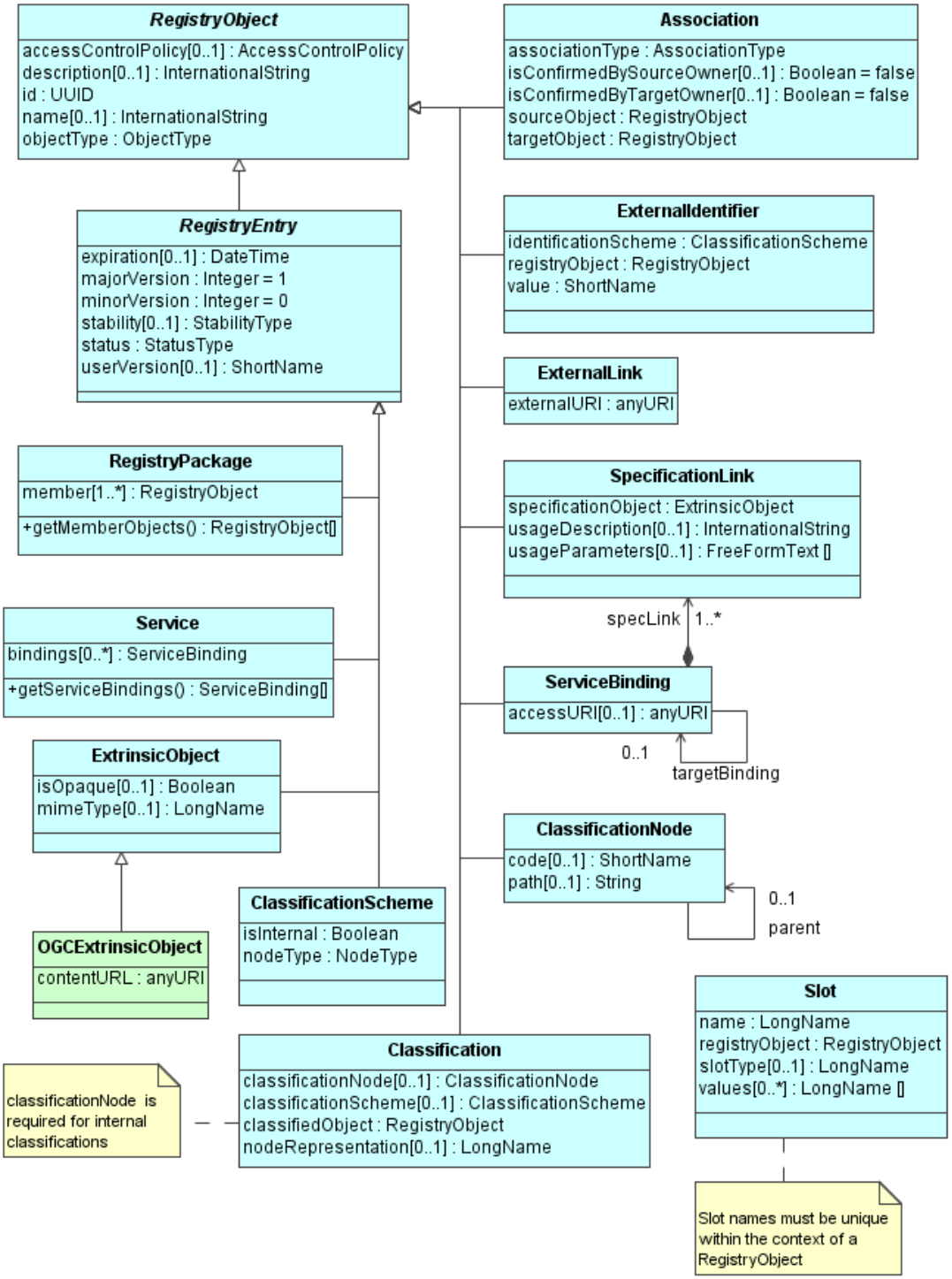
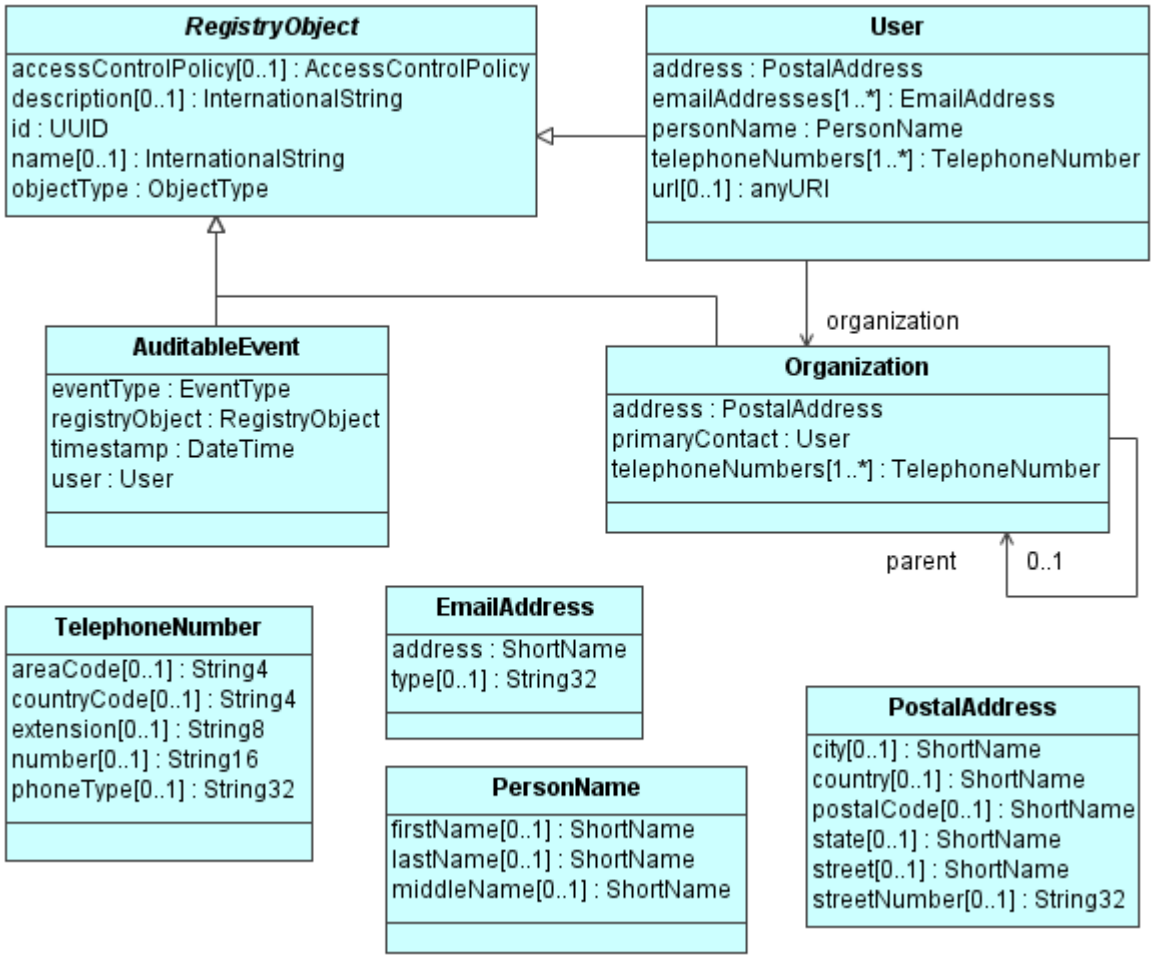


Figure 20 – Structure of Layer Metadata

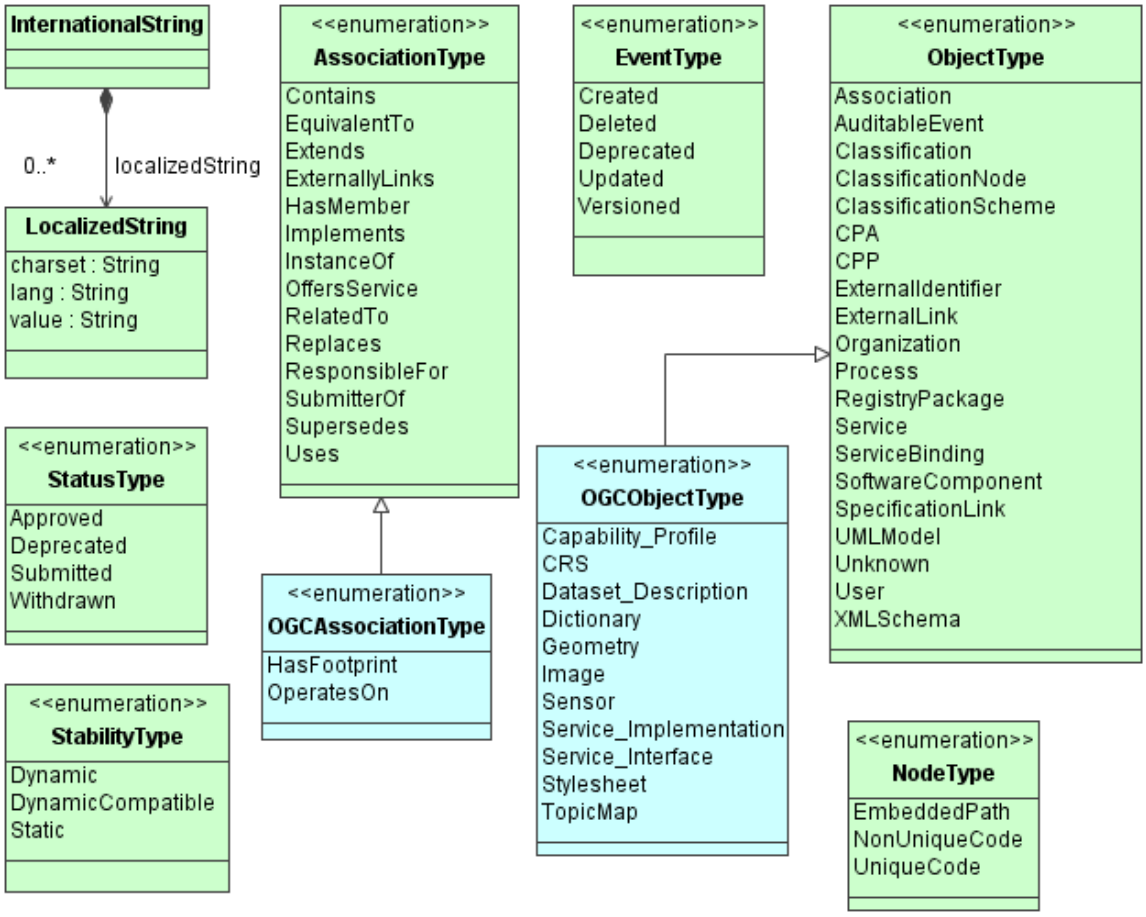
ANNEX A
ebRIM Core View (Informative)



ANNEX B
ebRIM Audit Trail View (Informative)



ANNEX C
Enumeration of Object and Association Types (Informative)



ANNEX D
Example CSW 2.0 XML-encoded Operations

For now this annex is mean to gather all the example registry queries into one location. Eventually, this annex will be edited and formatted to present the most useful queries from OWS3.

QUERY 1:

Get all the registry records for all feature and layer names of type "Roads". The query basically says:

Find all ExtrinsicObject records of type "Layer" or "FeatureType" that are associated with the registry record whose name is "Roads". The association type (or name) should be "IsOfType".

```
<GetRecords>
   <Query typeName="ExtrinsicObject=e1
                    ExtrinsicObject=e2
                    Association=a">
      <PropertyName>/e</PropertyName>
      <Constraint>
         <Filter>
            <And>
               <PropertyIsEqualTo>
                  <PropertyName>/e1/@id</PropertyName>
                  <PropertyName>/a/@sourceObject</PropertyName>
               </PropertyIsEqualTo>
               <PropertyIsEqualTo>
                  <PropertyName>/a/@assocationType</PropertyName>
                  <Literal>IsOfType</Literal>
               </PropertyIsEqualTo>
               <PropertyIsEqualTo>
                  <PropertyName>/a/@targetObject</PropertyName>
                  <PropertyName>/e2/@id</PropertyName>
               </PropertyIsEqualTo>
               <PropertyIsEqualTo>
         <PropertyName>/e2/Name/LocalizedString/@value</PropertyName>
                  <Literal>Roads</Literal>
               </PropertyIsEqualTo>
               <Or>
                  <PropertyIsEqualTo>
                     <PropertyName>/e1/@objectType</PropertyName>
                     <Literal>Layer</Literal>
                  </PropertyIsEqualTo>
                  <PropertyIsEqualTo>
                     <PropertyName>/e1/@objectType</PropertyName>
                     <Literal>FeatureType</Literal>
                  </PropertyIsEqualTo>
               </Or>
            </And>
         </Filter>
      </Constraint>
   </Query>
</GetRecords>
```

An example response record is:

```
        <rim:ExtrinsicObject
             id="urn:uuid:f112de1e5ed54416ba9092ac065e3088"
             objectType="Layer"
             majorVersion="1"
             minorVersion="0"
             stability="Static"
             status="Approved">
           <rim:Name>
              <rim:LocalizedString lang="en-US" value="ROADL_1M:Foundation"/>
           </rim:Name>
           <rim:Description>
              <rim:LocalizedString lang="en-US"
                   value="VRF Narrative Table for &quot;Roads&quot;:&lt;P&gt;The majority of
data contained within the road line feature class was derivedfrom the roads layer of Digital
Chart of the World, edition 1, July 1992.  Allschematic road connectors added in urbanized
areas for network ..."/>
           </rim:Description>
           <rim:Slot name="Title">
              <rim:ValueList>
                 <rim:Value>Roads</rim:Value>
              </rim:ValueList>
           </rim:Slot>
        </rim:ExtrinsicObject>
```

NOTE: The actual response comes in a response container defined in the CSW specification. I have removed it for the sake of clarity.

QUERY 2:

For each record from QUERY 1, find the service(s) that offer the particular Layer/FeatureType.

```
        <GetRecords>
           <Query typeName="Service=s Association=a">
              <PropertyName>/s</PropertyName>
              <Constraint>
                 <Filter>
                    <And>
                       <PropertyIsEqualTo>
                          <PropertyName>/s/@id</PropertyName>
                          <PropertyName>/a/@sourceObject</PropertyName>
                       </PropertyIsEqualTo>
                       <PropertyIsEqualTo>
                          <PropertyName>/a/@assocationType</PropertyName>
                          <Literal>Offers</Literal>
                       </PropertyIsEqualTo>
                          <PropertyName>/a/@targetObject</PropertyName>
                    <Literal>urn:uuid:f112de1e5ed54416ba9092ac065e3088</Literal>
                          </PropertyIsEqualTo>
                    </And>
                 </Filter>
              </Constraint>
           </Query>
        </GetRecords>
```

An example response is:

```
   <rim:Service rim="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5"
     id="urn:uuid:8717ebd07d4f41e7b74447f88d41947a"
     objectType="urn:uuid:52fc5536-c38f-4e89-b661-9664fa1f592f">
     <rim:Name>
       <rim:LocalizedString lang="en-US" value="WMS"/>
     </rim:Name>
     <rim:Description>
       <rim:LocalizedString lang="en-US"
```

```
          value="WMS-compliant cascading map server by CubeWerx Inc."/>
      </rim:Description>
      <rim:Slot name="LayerLimit">
        <rim:ValueList>
          <rim:Value>-1</rim:Value>
        </rim:ValueList>
      </rim:Slot>
      <rim:Slot name="MaxHeight">
        <rim:ValueList>
          <rim:Value>3000</rim:Value>
        </rim:ValueList>
      </rim:Slot>
      <rim:Slot name="MaxWidth">
        <rim:ValueList>
          <rim:Value>4000</rim:Value>
        </rim:ValueList>
      </rim:Slot>
      <rim:Slot name="OnlineResource">
        <rim:ValueList>
          <rim:Value>http://www.cubewerx.com/</rim:Value>
        </rim:ValueList>
      </rim:Slot>
      <rim:Slot name="Service Type">
        <rim:ValueList>
          <rim:Value>WMS</rim:Value>
        </rim:ValueList>
      </rim:Slot>
      <rim:Slot name="Title">
        <rim:ValueList>
          <rim:Value>CubeSERV</rim:Value>
        </rim:ValueList>
      </rim:Slot>
      <rim:Classification id="urn:uuid:afd8979c159144a69f767344c746d0a4"
        home="http://demo.cubewerx.com/cwwrs2/cwwrs.cgi" status="Approved"
        classificationScheme="urn:uuid:db5357c5-3d0d-4dfa-8735-770320727558"
        classifiedObject="urn:uuid:8717ebd07d4f41e7b74447f88d41947a"
        classificationNode="urn:uuid:3c7cb124-5dae-4b8c-87b5-
fc77fc8a3f4a"></rim:Classification>
      <rim:Classification id="urn:uuid:eafd10a775174d6d9ab2cf0a7be6deac"
        home="http://demo.cubewerx.com/cwwrs2/cwwrs.cgi" status="Approved"
        classificationScheme="urn:opengis:interop:ows:taxonomy-ebrim:services:v0.7.2"
        classifiedObject="urn:uuid:8717ebd07d4f41e7b74447f88d41947a"
        classificationNode="urn:opengis:interop:ows:taxonomy-
ebrim:services:v0.7.2:WMS"></rim:Classification>
      <rim:ServiceBinding id="urn:uuid:1c75e06129214c9bb943318094b0f94f"
        home="http://demo.cubewerx.com/cwwrs2/cwwrs.cgi" status="Approved"
        service="urn:uuid:8717ebd07d4f41e7b74447f88d41947a"
        accessURI="http://demo.cubewerx.com/demo/cubeserv/cubeserv.cgi">
        <rim:Name>
          <rim:LocalizedString lang="en-US" value="GetCapabilities"/>
        </rim:Name>
        <rim:Slot name="outputFormat">
          <rim:ValueList>
            <rim:Value>text/xml</rim:Value>
            <rim:Value>application/x-bxml; version="0.0.8"</rim:Value>
          </rim:ValueList>
        </rim:Slot>
      </rim:ServiceBinding>
      <rim:ServiceBinding id="urn:uuid:a125a0da65b54ba3881acba2ed6dc305"
        home="http://demo.cubewerx.com/cwwrs2/cwwrs.cgi" status="Approved"
        service="urn:uuid:8717ebd07d4f41e7b74447f88d41947a"
        accessURI="http://demo.cubewerx.com/demo/cubeserv/cubeserv.cgi?">
        <rim:Name>
          <rim:LocalizedString lang="en-US" value="GetCapabilities"/>
        </rim:Name>
        <rim:Slot name="outputFormat">
          <rim:ValueList>
            <rim:Value>text/xml</rim:Value>
            <rim:Value>application/x-bxml; version="0.0.8"</rim:Value>
          </rim:ValueList>
```

```
      </rim:Slot>
    </rim:ServiceBinding>
    <rim:ServiceBinding id="urn:uuid:5063461b2755475bb98e0f159818d880"
      home="http://demo.cubewerx.com/cwwrs2/cwwrs.cgi" status="Approved"
      service="urn:uuid:8717ebd07d4f41e7b74447f88d41947a"
      accessURI="http://demo.cubewerx.com/demo/cubeserv/cubeserv.cgi">
      <rim:Name>
        <rim:LocalizedString lang="en-US" value="GetMap"/>
      </rim:Name>
      <rim:Slot name="outputFormat">
        <rim:ValueList>
          <rim:Value>image/png</rim:Value>
          <rim:Value>image/png; PhotometricInterpretation=PaletteColor</rim:Value>
          <rim:Value>image/png; PhotometricInterpretation=RGB</rim:Value>
          <rim:Value>image/png</rim:Value>
          <rim:Value>image/tiff</rim:Value>
          <rim:Value>image/tiff; PhotometricInterpretation=RGB</rim:Value>
          <rim:Value>image/gif</rim:Value>
          <rim:Value>image/jpeg</rim:Value>
          <rim:Value>image/ppm</rim:Value>
          <rim:Value>application/x-cubestor-wkb</rim:Value>
          <rim:Value>application/x-cubestor-wkb;
PhotometricInterpretation=PaletteColor</rim:Value>
          <rim:Value>application/x-cubestor-wkb; PhotometricInterpretation=RGB</rim:Value>
          <rim:Value>application/x-cubestor-wkb;
PhotometricInterpretation=BlackIsZero</rim:Value>
          <rim:Value>text/xml; subtype="gml.1"</rim:Value>
        </rim:ValueList>
      </rim:Slot>
    </rim:ServiceBinding>
    <rim:ServiceBinding id="urn:uuid:eccd16e5ad2f45ee9b8d72b83cb3a4c6"
      home="http://demo.cubewerx.com/cwwrs2/cwwrs.cgi" status="Approved"
      service="urn:uuid:8717ebd07d4f41e7b74447f88d41947a"
      accessURI="http://demo.cubewerx.com/demo/cubeserv/cubeserv.cgi?">
      <rim:Name>
        <rim:LocalizedString lang="en-US" value="GetMap"/>
      </rim:Name>
      <rim:Slot name="outputFormat">
        <rim:ValueList>
          <rim:Value>image/png</rim:Value>
          <rim:Value>image/png; PhotometricInterpretation=PaletteColor</rim:Value>
          <rim:Value>image/png; PhotometricInterpretation=RGB</rim:Value>
          <rim:Value>image/png</rim:Value>
          <rim:Value>image/tiff</rim:Value>
          <rim:Value>image/tiff; PhotometricInterpretation=RGB</rim:Value>
          <rim:Value>image/gif</rim:Value>
          <rim:Value>image/jpeg</rim:Value>
          <rim:Value>image/ppm</rim:Value>
          <rim:Value>application/x-cubestor-wkb</rim:Value>
          <rim:Value>application/x-cubestor-wkb;
PhotometricInterpretation=PaletteColor</rim:Value>
          <rim:Value>application/x-cubestor-wkb; PhotometricInterpretation=RGB</rim:Value>
          <rim:Value>application/x-cubestor-wkb;
PhotometricInterpretation=BlackIsZero</rim:Value>
          <rim:Value>text/xml; subtype="gml.1"</rim:Value>
        </rim:ValueList>
      </rim:Slot>
    </rim:ServiceBinding>
    <rim:ServiceBinding id="urn:uuid:7f40c6b5e4b2435aa6ea6142c6513873"
      home="http://demo.cubewerx.com/cwwrs2/cwwrs.cgi" status="Approved"
      service="urn:uuid:8717ebd07d4f41e7b74447f88d41947a"
      accessURI="http://demo.cubewerx.com/demo/cubeserv/cubeserv.cgi">
      <rim:Name>
        <rim:LocalizedString lang="en-US" value="GetFeatureInfo"/>
      </rim:Name>
      <rim:Slot name="outputFormat">
        <rim:ValueList>
          <rim:Value>text/html</rim:Value>
          <rim:Value>text/xml; subtype="gml.1"</rim:Value>
          <rim:Value>application/x-bxml; version="0.0.8"; subtype="gml.1"</rim:Value>
```

```
        <rim:Value>text/xml; subtype="bxfs/0.0.1"</rim:Value>
        <rim:Value>application/x-bxml; version="0.0.8"; subtype="bxfs/0.0.1"</rim:Value>
        <rim:Value>application/x-cubestor-any</rim:Value>
      </rim:ValueList>
    </rim:Slot>
  </rim:ServiceBinding>
  <rim:ServiceBinding id="urn:uuid:cc442d7202d54b3a8c3b241389a4c880"
    home="http://demo.cubewerx.com/cwwrs2/cwwrs.cgi" status="Approved"
    service="urn:uuid:8717ebd07d4f41e7b74447f88d41947a"
    accessURI="http://demo.cubewerx.com/demo/cubeserv/cubeserv.cgi?">
    <rim:Name>
      <rim:LocalizedString lang="en-US" value="GetFeatureInfo"/>
    </rim:Name>
    <rim:Slot name="outputFormat">
      <rim:ValueList>
        <rim:Value>text/html</rim:Value>
        <rim:Value>text/xml; subtype="gml.1"</rim:Value>
        <rim:Value>application/x-bxml; version="0.0.8"; subtype="gml.1"</rim:Value>
        <rim:Value>text/xml; subtype="bxfs/0.0.1"</rim:Value>
        <rim:Value>application/x-bxml; version="0.0.8"; subtype="bxfs/0.0.1"</rim:Value>
        <rim:Value>application/x-cubestor-any</rim:Value>
      </rim:ValueList>
    </rim:Slot>
  </rim:ServiceBinding>
  <rim:ServiceBinding id="urn:uuid:99dde3c2684348cd96513142ee0bedfe"
    home="http://demo.cubewerx.com/cwwrs2/cwwrs.cgi" status="Approved"
    service="urn:uuid:8717ebd07d4f41e7b74447f88d41947a"
    accessURI="http://demo.cubewerx.com/demo/cubeserv/cubeserv.cgi">
    <rim:Name>
      <rim:LocalizedString lang="en-US" value="TransformGeometry"/>
    </rim:Name>
    <rim:Slot name="outputFormat">
      <rim:ValueList>
        <rim:Value>text/xml</rim:Value>
        <rim:Value>application/x-bxml; version="0.0.8"</rim:Value>
      </rim:ValueList>
    </rim:Slot>
  </rim:ServiceBinding>
 <rim:ServiceBinding id="urn:uuid:cd1712514030416ba1734e1144601a83"
    home="http://demo.cubewerx.com/cwwrs2/cwwrs.cgi" status="Approved"
    service="urn:uuid:8717ebd07d4f41e7b74447f88d41947a"
    accessURI="http://demo.cubewerx.com/demo/cubeserv/cubeserv.cgi?">
    <rim:Name>
      <rim:LocalizedString lang="en-US" value="TransformGeometry"/>
    </rim:Name>
    <rim:Slot name="outputFormat">
      <rim:ValueList>
        <rim:Value>text/xml</rim:Value>
        <rim:Value>application/x-bxml; version="0.0.8"</rim:Value>
      </rim:ValueList>
    </rim:Slot>
  </rim:ServiceBinding>
  <rim:ServiceBinding id="urn:uuid:212bb46b15bb4810acaf7b0c4255466c"
    home="http://demo.cubewerx.com/cwwrs2/cwwrs.cgi" status="Approved"
    service="urn:uuid:8717ebd07d4f41e7b74447f88d41947a"
    accessURI="http://demo.cubewerx.com/demo/cubeserv/cubeserv.cgi?">
    <rim:Name>
      <rim:LocalizedString lang="en-US" value="DescribeLayer"/>
    </rim:Name>
    <rim:Slot name="outputFormat">
      <rim:ValueList>
        <rim:Value>text/xml</rim:Value>
      </rim:ValueList>
    </rim:Slot>
  </rim:ServiceBinding>
  <rim:ServiceBinding id="urn:uuid:d0a3ea6e0e54434dabe6e22c846d7c78"
    home="http://demo.cubewerx.com/cwwrs2/cwwrs.cgi" status="Approved"
    service="urn:uuid:8717ebd07d4f41e7b74447f88d41947a"
    accessURI="http://demo.cubewerx.com/demo/cubeserv/cubeserv.cgi?">
    <rim:Name>
```

```
        <rim:LocalizedString lang="en-US" value="GetLegendGraphic"/>
      </rim:Name>
      <rim:Slot name="outputFormat">
        <rim:ValueList>
          <rim:Value>image/png</rim:Value>
          <rim:Value>image/png; PhotometricInterpretation=PaletteColor</rim:Value>
          <rim:Value>image/png; PhotometricInterpretation=RGB</rim:Value>
          <rim:Value>image/png</rim:Value>
          <rim:Value>image/tiff</rim:Value>
          <rim:Value>image/tiff; PhotometricInterpretation=RGB</rim:Value>
          <rim:Value>image/gif</rim:Value>
          <rim:Value>image/jpeg</rim:Value>
          <rim:Value>image/ppm</rim:Value>
          <rim:Value>application/x-cubestor-wkb</rim:Value>
          <rim:Value>application/x-cubestor-wkb;
PhotometricInterpretation=PaletteColor</rim:Value>
          <rim:Value>application/x-cubestor-wkb; PhotometricInterpretation=RGB</rim:Value>
          <rim:Value>application/x-cubestor-wkb;
PhotometricInterpretation=BlackIsZero</rim:Value>
        </rim:ValueList>
      </rim:Slot>
    </rim:ServiceBinding>
    <rim:ServiceBinding id="urn:uuid:4204eaa49f6d4416919a45b62fc823d1"
      home="http://demo.cubewerx.com/cwwrs2/cwwrs.cgi" status="Approved"
      service="urn:uuid:8717ebd07d4f41e7b74447f88d41947a"
      accessURI="http://demo.cubewerx.com/demo/cubeserv/cubeserv.cgi?">
      <rim:Name>
        <rim:LocalizedString lang="en-US" value="GetLegend"/>
      </rim:Name>
      <rim:Slot name="outputFormat">
        <rim:ValueList>
          <rim:Value>image/png</rim:Value>
          <rim:Value>image/png; PhotometricInterpretation=PaletteColor</rim:Value>
          <rim:Value>image/png; PhotometricInterpretation=RGB</rim:Value>
          <rim:Value>image/png</rim:Value>
          <rim:Value>image/tiff</rim:Value>
          <rim:Value>image/tiff; PhotometricInterpretation=RGB</rim:Value>
          <rim:Value>image/gif</rim:Value>
          <rim:Value>image/jpeg</rim:Value>
          <rim:Value>image/ppm</rim:Value>
          <rim:Value>application/x-cubestor-wkb</rim:Value>
          <rim:Value>application/x-cubestor-wkb;
PhotometricInterpretation=PaletteColor</rim:Value>
          <rim:Value>application/x-cubestor-wkb; PhotometricInterpretation=RGB</rim:Value>
          <rim:Value>application/x-cubestor-wkb;
PhotometricInterpretation=BlackIsZero</rim:Value>
        </rim:ValueList>
      </rim:Slot>
    </rim:ServiceBinding>
    <rim:ServiceBinding
         id="urn:uuid:852996315f1d474da30ec066aac74c17"
         home="http://demo.cubewerx.com/cwwrs2/cwwrs.cgi"
         status="Approved"
         service="urn:uuid:8717ebd07d4f41e7b74447f88d41947a"
         accessURI="http://demo.cubewerx.com/demo/cubeserv/cubeserv.cgi?">
      <rim:Name>
        <rim:LocalizedString lang="en-US" value="GetStyles"/>
      </rim:Name>
      <rim:Slot name="outputFormat">
        <rim:ValueList>
          <rim:Value>text/xml</rim:Value>
          <rim:Value>application/x-bxml; version="0.0.8"</rim:Value>
        </rim:ValueList>
      </rim:Slot>
    </rim:ServiceBinding>
  </rim:Service>
```

ASSUMPTIONS:

The following assumptions where made:

1. There exists an association in the registry, called 'Offers', between a service and the data that the service offers. This may be created manually or automatically by the server during the harvest process.

2. There desired feature type, 'Roads' in this example, is a registered type.

3. An association exists, in the Association entity of the ebRIM, called 'IsOfType' between the registered feature type (2) and each layer/feature of that type harvested in (1). Such an association may be created manually using the Transaction request.

GeoDSS Query Examples

The following clause contains a representative set of queries that were executable against the catalogs during the OWS3 project. Each query is expressed using the CSW 2.0 API and is targeted against both the ebRIM and OGC Core information models.

The next 4 sections discuss each query and present examples. A narrative describing the query is provided and then each query encoding is presented and discussed.

Response documents are not presented because they are too long.

Property Mappings

The following table declares the mapping between the OGC core elements and the ebRIM that were used to formulate the example queries in this section.

```
Property    EBRIM (everything is in the rim namespace)
----------  ---------------------------------------------------
Subject     /ExtrinsicObject/Name/LocalizedString/@value
Title       /ExtrinsicObject/Slot[@name="title"]/ValueList/Value[1]
Modified    /ExtrinsicObject/Slot[@name="dateModified"]/ValueList/Value[1]
Creator     /ExtrinsicObject/Slot[@name="creator"]/ValueList/Value[1]
Publisher   /ExtrinsicObject/Slot[@name="publisher"]/ValueList/Value[1]
ObjectType  /ExtrinsicObject/@objectType
Geometry    /ExtrinsicObject/Slot[@name="geometry"]/ValueList/Value[1]
Security    /ExtrinsicObject/Slot[@name="security attribute name"]/ValueList/Value[1]

Property   csw:Record
---------- ---------------------------------------------------
Subject    /csw:Record/dc:subject
Title      /csw:Record/dc:title
Modified   /csw:Record/dc:modified
Creator    /csw:Record/dc:creator
Publisher  /csw:Record/dc:publisher
ObjectType /csw:Record/dc:type
Geometry   /csw:Record/dc:coverage/dc:spatial <- *** must be gml:Envelope
Security   /csw:Record/myns:SecurityLevel (extension for OWS3)
```

Query 1:

A typical query against a catalog to search for UML Models GML Application Schemas will utilize the properties:

Subject (i.e. a theme name or a product name)

Title

Modified (Date)

Creator or Publisher

These queries could be either a single property query (find all NGA) but may also be a multiple property query. For example, Get all Subject=(MSD1) and Creator=(NGA) and Security=(U).

The following queries either have a single predicate or use the **ogc:And** logical operator to combine predicates in order to present "single property" queries and "multiple property" queries.

In all cases, a predicate is included that restricts the query to GML Application Schemas or UML Model.

```
EBRIM Query:
   <GetRecords
      xmlns="http://www.opengis.net/cat/csw"
      xmlns:rim="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5"
      xmlns:ogc="http://www.opengis.net/ogc"
      outputFormat="text/xml; charset=UTF-8">
      <Query typeNames="rim:ExtrinsicObject">
         <Constraint>
            <ogc:Filter>
               <ogc:And>
                  <ogc:PropertyIsEqualTo>
                     <ogc:PropertyName>/rim:ExtrinsicObject/@objectType</ogc:PropertyName>
                     <ogc:Literal>GML Application Schema</ogc:Literal>
                  </ogc:PropertyIsEqualTo>
                  <ogc:PropertyIsEqualTo>

<ogc:PropertyName>/rim:ExtrinsicObject/rim:Slot[@name="creator"]/rim:ValueList/rim:Value[1]</
ogc:PropertyName>
                     <ogc:Literal>NGA</ogc:Literal>
                  </ogc:PropertyIsEqualTo>
               <ogc:And>
            </ogc:Filter>
         </Constraint>
      </Query>
   </GetRecords>

CORE Query:
   <GetRecords
      xmlns="http://www.opengis.net/cat/csw"
      xmlns:ogc="http://www.opengis.net/ogc"
      xmlns:dc="http://www.purl.org/dc/elements/1.1/"
      outputFormat="text/xml; charset=UTF-8">
      <Query typeNames="Record">
         <Constraint>
            <ogc:Filter>
               <ogc:And>
                  <ogc:PropertyIsEqualTo>
                     <ogc:PropertyName>/csw:Record/dc>type</ogc:PropertyName>
                     <ogc:Literal>UML Model</ogc:Literal>
                  <ogc:PropertyIsEqualTo>
                  <ogc:PropertyIsEqualTo>
                     <ogc:PropertyName>/Record/dc:creator</ogc:PropertyName>
                     <ogc:Literal>NGA</ogc:Literal>
                  </ogc:PropertyIsEqualTo>
```

```
            </ogc:And>
         </ogc:Filter>
      </Constraint>
   </Query>
</GetRecords>

EBRIM Query:
<GetRecords
   xmlns="http://www.opengis.net/cat/csw"
   xmlns:rim="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5"
   xmlns:ogc="http://www.opengis.net/ogc"
   outputFormat="text/xml; charset=UTF-8">
   <Query typeNames="rim:ExtrinsicObject">
      <Constraint>
         <ogc:Filter>
            <ogc:And>
               <ogc:PropertyIsEqualTo>
                  <ogc:PropertyName>/rim:ExtrinsicObject/@objectType</ogc:PropertyName>
                  <ogc:Literal>UML Model</ogc:Literal>
               <ogc:PropertyIsEqualTo>
               <ogc:PropertyIsEqualTo>

<ogc:PropertyName>/rim:ExtrinsicObject/rim:Slot[@name="creator"]/rim:ValueList/rim:Value[1]</
ogc:PropertyName>
                  <ogc:Literal>NGA</ogc:Literal>
               </ogc:PropertyIsEqualTo>
               <ogc:PropertyIsEqualTo>

<ogc:PropertyName>/rim:ExtrinsicObject/rim:Name/rim:LocalizedString/@value</ogc:PropertyName>
                  <ogc:Literal>MSD1</ogc:Literal>
               </ogc:PropertyIsEqualTo>
               <ogc:PropertyIsEqualTo>

<ogc:PropertyName>/rim:ExtrinsicObject/rim:Slot[@name="security"]/rim:ValueList/rim:Value[1]<
/ogc:PropertyName>
                  <ogc:Literal>U</ogc:Literal>
               </ogc:PropertyIsEqualTo>
            </ogc:And>
         </ogc:Filter>
      </Constraint>
   </Query>
</GetRecords>

CORE Query:
<GetRecords
   xmlns="http://www.opengis.net/cat/csw"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:dc="http://www.purl.org/dc/elements/1.1/"
   outputFormat="text/xml; charset=UTF-8">
   <Query typeNames="Record">
      <Constraint>
         <ogc:Filter>
            <ogc:And>
               <ogc:PropertyIsEqualTo>
                  <ogc:PropertyName>/csw:Record/dc>type</ogc:PropertyName>
                  <ogc:Literal>GML Application Schema</ogc:Literal>
               <ogc:PropertyIsEqualTo>
                  <ogc:PropertyName>/Record/dc:creator</ogc:PropertyName>
                  <ogc:Literal>NGA</ogc:Literal>
               </ogc:PropertyIsEqualTo>
               <ogc:PropertyIsEqualTo>
                  <ogc:PropertyName>/Record/dc:subject</ogc:PropertyName>
                  <ogc:Literal>MSD1</ogc:Literal>
               </ogc:PropertyIsEqualTo>
               <ogc:PropertyIsEqualTo>
                  <ogc:PropertyName>/Record/myns:SecurityLevel</ogc:PropertyName>
                  <ogc:Literal>U</ogc:Literal>
               </ogc:PropertyIsEqualTo>
            </ogc:And>
         </ogc:Filter>
```

```
         </Constraint>
      </Query>
   </GetRecords>
```

Query 2:

A query may also wish to differentiate based on either GML Application Schema or UML model.

The type of an object being registered can be determined a number of ways including using the mime-type of the resource being registered or being set specifically by the entity (person/program) registering the resource.

Both the /csw:Records/dc:type and /rim:ExtrinsicObject/@objectType properties from each information model allow this differentiation to be made once a resource has been registered.

The following predicates restrict the queries to the objects of interest (GML app schemas and UML models):

```
EBRIM:
      <PropertyIsEqual>
         <PropertyName>/rim:ExtrinsicObject/@objectType</PropertyName>
         <Literal>GML Application Schema</Literal>
      </PropertyIsEqual>
      -- OR --
      <PropertyIsEqual>
         <PropertyName>/rim:ExtrinsicObject/@objectType</PropertyName>
         <Literal>UML Model</Literal>
      </PropertyIsEqual>

CORE:
      <PropertyIsEqual>
         <PropertyName>/csw:Record/dc:type</PropertyName>
         <Literal>GML Application Schema</Literal>
      </PropertyIsEqual>
      -- OR --
      <PropertyIsEqual>
         <PropertyName>/csw:Record/dc:type</PropertyName>
         <Literal>UML Model</Literal>
      </PropertyIsEqual>
```

Query 3

A query for data may require an additional set of catalog metadata but could use Subject, Date, Envelope (Geospatial Coverage and Temporal)

The following example query looks for "datasets" in some area of interest, specified using the gml:Envelope element:

```
   <GetRecords
      xmlns="http://www.opengis.net/cat/csw"
      xmlns:rim="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5"
      xmlns:gml="http://www.opengis.org/gml"
      xmlns:ogc="http://www.opengis.net/ogc"
      outputFormat="text/xml; charset=UTF-8">
      <Query typeNames="rim:ExtrinsicObject">
         <Constraint>
```

```
            <ogc:Filter>
               <ogc:And>
                  <PropertyIsEqual>
                     <PropertyName>/rim:ExtrinsicObject/@objectType</PropertyName>
                     <Literal>Dataset</Literal>
                  </PropertyIsEqual>
                  <ogc:BBOX>

<ogc:PropertyName>/rim:ExtrinsicObject/rim:Slot[@name="geometry"]/rim:ValueList/rim:Value[1]<
/ogc:PropertyName>
                     <gml:Envelope> ... </gml:Envelope>
                  </ogc:BBOX>
               <ogc:And>
            </ogc:Filter>
         </Constraint>
      </Query>
   </GetRecords>
```

Some comments about TEMPORAL queries...

Queries can be temporally constrained using the standard scalar operators found in the Filter specification (PropertyIsBetween, PropertyIsEqualTo, etc...).

However, the Filter specification does not formally define temporal operators although spatial operators such as Inside, Overlaps, etc could be overloaded for the task. The CSW 2.0 specification does define the temporal operations **Before**, **During** and **After** which could easily be added to the Filter Encoding Specification (as **PropertyIsBefore**, **PropertyIsDuring** and **PropertyIsAfter**).

Here is an example of a temporal query finding records modified between two specific dates that uses scalar operators to constrain the "modified" property (which is assumed to be of datatype date):

```
   <GetRecords
      xmlns="http://www.opengis.net/cat/csw"
      xmlns:rim="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5"
      xmlns:ogc="http://www.opengis.net/ogc"
      outputFormat="text/xml; charset=UTF-8">
      <Query typeNames="rim:ExtrinsicObject">
         <Constraint>
            <ogc:Filter>
               <ogc:PropertyIsBetween>

<ogc:PropertyName>/rim:ExtrinsicObject/rim:Slot[@name="modified"]/rim:ValueList/rim:Value[1]<
/ogc:PropertyName>
                  <ogc:LowerBoundary>
                     <ogc:Literal>2005-05-24T08-00:00-05:00</ogc:Literal>
                  </ogc:LowerBoundary>
                  <ogc:UpperBoundary>
                     <ogc:Literal>2005-05-24T08:30:00-05:00</ogc:Literal>
                  </ogc:UpperBoundary>
               </ogc:PropertyIsBetween>
            </ogc:Filter>
         </Constraint>
      </Query>
   </GetRecords>
```

Query 4:

This query example is based more on query for data and not query on App Sch or UML. But it does raise the question as to whether the information we provide to the catalog team shouldn't cover both use cases (data and schemas). This example also brings up the question of what queryable property would contain the word "Iraq", subject?

Example: RETRIEVE [any] "MSD3" OVERLAPPING "Iraq"

There are at least two solutions to this query:

Solution 1 involves a two-step procedure. Step one is to query a gazetteer using the keyword "Iraq" to get the spatial extent of the country. Then the following query could be executed on the catalog:

```
   <GetRecords
      xmlns="http://www.opengis.net/cat/csw"
      xmlns:rim="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5"
      xmlns:gml="http://www.opengis.org/gml"
      xmlns:ogc="http://www.opengis.net/ogc"
      outputFormat="text/xml; charset=UTF-8">
      <Query typeNames="rim:ExtrinsicObject">
         <Constraint>
            <ogc:Filter>
               <ogc:And>
                  <ogc:PropertyIsEqualTo>

<ogc:PropertyName>/ExtrinsicObject/Name/LocalizedString/@value</ogc:PropertyName>
                     <ogc:Literal>MSD3</ogc:Literal>
                  </ogc:PropertyIsEqualTo>
                  <ogc:Overlaps>

<ogc:PropertyName>/rim:ExtrinsicObject/rim:Slot[@name="geometry"]/rim:ValueList/rim:Value[1]<
/ogc:PropertyName>
                     <gml:Polygon>
                        <gml:exterior>
                           <gml:LinearRing>
                           ...
                           </gml:LinearRing>
                        </gml:exterior>
                     </gml:Polygon>
                  </ogc:Overlaps>
               </ogc:And>
            </ogc:Filter>
         </Constraint>
      </Query>
   </GetRecords>
```

Solution 2 involves using a geographic taxonomy. Typically, such taxonomies have the following hierarchy:

```
   World -> Continent -> Country -> AdminArea -> City ...
```

You can imagine that such a taxonomy for Africa might look like this:

```
   Africa
   |
   +-----+-----+----- ...
   |     |     |
   Iraq   Iran   South Africa
   |
```

```
   |
   |
   +---+---+--- ...
   |   |   |
   "I don't know the provinces of Iraq"
   |
   |
   +--------+--------+---- ...
   |        |        |
   Baghdad  Tikrit   Mosul
```

Each record in the catalog would need to be classified using such a taxonomy. You can then find all catalog records that are classified as "Iraq" by finding all records classified as "Iraq" or classified using any child node of "Iraq".

In other words each node in the hierarchy acts as an aggregation point. So if the node is "Iraq" you get all Iraq related records.

Resolving such a query involves a number of steps on the client side. Roughly speaking, a client would need to execute the following set of operations:

```
Step 1:  Cache the classification scheme on the client by executing the
         following CSW query that would return an XML document containing
         all the nodes in the classification scheme.

         <GetRecords
            xmlns="http://www.opengis.net/cat/csw"
            xmlns:rim="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5"
            xmlns:gml="http://www.opengis.org/gml"
            xmlns:ogc="http://www.opengis.net/ogc"
            outputFormat="text/xml; charset=UTF-8">
            <Query typeNames="rim:ClassificationScheme">
               <Constraint>
                  <ogc:Filter>
                     <ogc:PropertyIsEqualTo>

<ogc:PropertyName>/rim:ClassificationScheme/Name/LocalizedString/@value</ogc:PropertyName>
                        <ogc:Literal>Some GeoTaxonomy Name</ogc:Literal>
                     </ogc:PropertyIsEqualTo>
                  </ogc:Filter>
               </Constraint>
            </Query>
         </GetRecords>

Step 2: Process the resulting XML document from "Step 1" and find the id's
        of all the descendants of the node of interest.  In this case,
        the node of interest would be the node with the code value of "Iraq".
        I believe a XPath expression could return the required information.

Step 3: Execute the following query.  The <ogc:Or> operator would
        have to contain one <PropertyIsEqualTo> predicate for each id
        from "Step 2".

        <GetRecords
           xmlns="http://www.opengis.net/cat/csw"
           xmlns:rim="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5"
           xmlns:gml="http://www.opengis.org/gml"
           xmlns:ogc="http://www.opengis.net/ogc"
           outputFormat="text/xml; charset=UTF-8">
           <Query typeNames="rim: ExtrinsicObject rim:ClassificationNode">
              <Constraint>
                 <ogc:Filter>
                    <ogc:And>
                       <ogc:PropertyIsEqualTo>
```

```
<ogc:PropertyName>/rim:ExtrinsicObject/@id</ogc:PropertyName>
<ogc:Literal>/rim:ClassificationNode/@classifiedObject</ogc:Literal>
                       </ogc:PropertyIsEqualTo>
                       <ogc:Or>
                          <ogc:PropertyIsEqualTo>
<ogc:PropertyName>/rim:ClassificationNode/@id</ogc:PropertyName>
<ogc:Literal>[an id from "Step 2"]</ogc:Literal>
                          </ogc:PropertyIsEqualTo>
                          <ogc:PropertyIsEqualTo>
<ogc:PropertyName>/rim:ClassificationNode/@id</ogc:PropertyName>
<ogc:Literal>[an id from "Step 2"]</ogc:Literal>
                          </ogc:PropertyIsEqualTo>
                          <ogc:PropertyIsEqualTo>
<ogc:PropertyName>/rim:ClassificationNode/@id</ogc:PropertyName>
<ogc:Literal>[an id from "Step 2"]</ogc:Literal>
                          </ogc:PropertyIsEqualTo>
                          <ogc:PropertyIsEqualTo>
<ogc:PropertyName>/rim:ClassificationNode/@id</ogc:PropertyName>
<ogc:Literal>[an id from "Step 2"]</ogc:Literal>
                          </ogc:PropertyIsEqualTo>
                          .
                          .
                          .
                       <ogc:Or>
                    </ogc:And>
                 </ogc:Filter>
              </Constraint>
           </Query>
        </GetRecords>
```

ANNEX E
Issues & Extensions

1. Filter specification needs to be extended allowed XPath expressions
   that more easily support ebRIM slots. Specifically, Filter needs to
   support something like:

```
                              this bit here
                              vvvvvvvvvvvvvvvvvv
/rim:ExtrinsicObject/rim:Slot[@name="geometry"]/rim:ValueList/rim:Value[1]
```

   This needs to be discussed further.

2. The current method of doing spatial queries with EBRIM involves
   an additional registry object (added by OGC) called "Geometry". A
   spatial query thus requires a fairly complex join between ExtrinsicObject,
   Geometry and Association to do a spatial query. Here is an example:

```
<GetRecords
   xmlns="http://www.opengis.net/cat/csw"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:gml="http://www.opengis.net/gml"
   elementSet="full"
   startPosition="1"
   maxRecords="5"
   outputFormat="application/xml; charset=UTF-8">
   <Query typeNames="rim:ExtrinsicObject__eo
                     rim:Association__a2
                     rim:Geometry__g">
      <ogc:ElementName>/eo</ogc:ElementName>
      <Constraint>
         <ogc:Filter>
            <ogc:And>
               <ogc:PropertyIsEqualTo>
                  <ogc:PropertyName>/a2/@associationType</ogc:PropertyName>
                  <ogc:Literal>HasFootprint</ogc:Literal>
               </ogc:PropertyIsEqualTo>
               <ogc:PropertyIsEqualTo>
                  <ogc:PropertyName>/a2/@sourceObject</ogc:PropertyName>
                  <ogc:PropertyName>/eo/@id</ogc:PropertyName>
               </ogc:PropertyIsEqualTo>
               <ogc:PropertyIsEqualTo>
                  <ogc:PropertyName>/a2/@targetObject</ogc:PropertyName>
                  <ogc:PropertyName>/g/@parent</ogc:PropertyName>
               </ogc:PropertyIsEqualTo>
               <ogc:BBOX>
                  <ogc:PropertyName>/g/content</ogc:PropertyName>
                  <gml:Box>
                     <gml:coordinates>-10,-10 -10,-10</gml:coordinates>
                  </gml:Box>
               </ogc:BBOX>
            </ogc:And>
         </ogc:Filter>
      </Constraint>
   </Query>
</GetRecords>
```

   In the examples in this note, I used a new proposal that Jerome and I
   discussed at the kickoff that really simplifies things. The proposal
   is simply to use a SLOT rather than define an entirely new registry
   object and rely on joins through the Association object.

   Thus the reference to the geometry of any object would be:

```
     /ExtrinsicObject/Slot[@name="geometry"]/ValueList/Value[1]
```

The issue here is that ebRIM does not allow "complex" or even XML values for SLOT values which it currently defines as strings of up to 128 characters.

We would need to change this definition to allow either a string or a gml geometry (i.e. XML fragment).

This needs to be discussed further.

2. There is no security property in the core properties or even the newly proposed core properties.

In my examples here, I assume that I have changed the definition of csw:Record

from:

```
<xsd:element name="Record" type="csw:RecordType"
             substitutionGroup="csw:AbstractRecord" />
<xsd:complexType name="RecordType" final="#all">
   <xsd:complexContent>
      <xsd:extension base="csw:DCMIRecordType">
         <xsd:sequence>
            <xsd:element ref="ows:BoundingBox" minOccurs="0" />
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

to:

```
<xsd:element name="Record" type="csw:RecordType"
             substitutionGroup="csw:AbstractRecord" />
<xsd:complexType name="RecordType" final="#all">
   <xsd:complexContent>
      <xsd:extension base="csw:DCMIRecordType">
         <xsd:sequence>
            <xsd:element ref="ows:BoundingBox"
                         minOccurs="0" />
            <xsd:element name="SecurityLevel"
                         type="xsd:string"
                         minOccurs="0" />
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

thus adding the SecurityLevel element. I think a better approach would be to have some sort of dynamically extensible core property set definition with the default definition being the current set of core properties. I am working on this.

This issue needs to be discussed.

3. Classification Queries

Clearly resolving classification queries is a pain due to all the joins! The abstract query definition in the CSW specification includes the definition of a classification operator which should be added to the Filter specification.

The following XML fragment shows what a ClassifiedAs operator might look like:

```
<ClassifiedAs scope="narrow">
   <TypeName>ExtrinsicObject</TypeName>
   <ClassificationScheme>urn:uuid:…</ClassificationScheme>
   <ClassificationNode>urn:uuid:…</ClassificationNode>
</ClassifiedAs>
```

```
The scope attribute is used to control how to classification tree is traversed.
The value "narrow" is used to mean all records classified with the specified node
and all its childer.  The value "broad" is used to mean all records classified
with the specified node and all its ancestors.  The value "exact" means all records
classified exactly with the specified node.

The TypeName element is used to specified the target of the classification operation.
In this example, all "ExtrinsicObject" records classified with the specified node.

The ClassificationScheme and ClassificationNode elements are used to identify the node
of interest.
```