Open Geospatial Consortium Inc.

Date: 2006-01-20

Reference number of this OGC® document: OGC 05-025r3

Version: 1.0.0

Category: OGC® Application Profile

Editor: R. Martell

OpenGIS® Catalogue Services — ebRIM (ISO/TS 15000-3) profile of CSW

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® Application Profile
Document subtype:	none
Document stage:	Draft proposed version
Document language:	English

Contents

Page

Figures

Page

Tables

Page

i. Preface

The OGC Catalogue Services 2.0 specification (OGC 04-021r3) establishes a general framework for implementing catalogue services that can be applied to meet the needs of stakeholders in a wide variety of domains. This application profile is based on the HTTP protocol binding described in Clause 10 of the Catalogue 2.0 specification; it qualifies as a 'Class 2' profile under the terms of ISO 19106 since it includes extensions permitted within the context of the base specifications, some of which are not part of the ISO 19100 series of geomatics standards.

ii. Document terms and definitions

This document uses the specification terms defined in Subclause 5.3 of [OGC 05-008], which is based on the ISO/IEC Directives, Part 2. Rules for the structure and drafting of International Standards. In particular, the word "shall" (not "must") is the verb form used to indicate a requirement to be strictly followed to conform to this specification.

iii. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Name	Organization
D. Androsevic <dandrosevicATgaldosincDOTcom>	Galdos Systems, Inc.
R. Martell (Editor) <rmartellATgaldosincDOTcom>	Galdos Systems, Inc.
J. Sonnet <jerome.sonnetATionicsoftDOTcom>	IONIC Software
L. Stainsby <lstainsbyATgaldosincDOTcom>	Galdos Systems, Inc
P. Vretanos <pvretanoATcubewerxDOTcom>	CubeWerx Inc.

iv. Revision history

Date	Release	Editor	Primary clauses modified	Description
2005-08-11	0.10.1	R. Martell	All	Adopted latest OGC template.
2005-09-15	0.10.2	R. Martell		
2005-10-17	0.10.3	R. Martell	10.2, Annex A	Candidate 1.0 release

v. Changes to the OGC Abstract Specification

The OpenGIS® Abstract Specification does not require any changes to accommodate the technical content of this document.

Foreword

This revision cancels and replaces OGC document 05-025. It depends primarily on the following base standards and specifications:

- OGC Catalogue Services Specification 2.0, with Technical Corrigendum 1 (OGC 04-021r3)
- OWS Common Implementation Specification 1.0 (OGC 05-008)
- Filter Encoding Implementation Specification 1.1 (OGC 04-095)
- OASIS ebXML Registry Information Model v3.0[1]
- IETF RFC 2616 (Hypertext Transfer Protocol -- HTTP/1.1)

This document includes four annexes. Annexes A and B are normative; all others are informative.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The OGC shall not be held responsible for identifying any or all such patent rights.

[1] The ebXML registry information model (ebRIM) is also published as ISO/TS 15000-3.

Introduction

The target audience for this document includes client and service developers, system testers, and users who want to acquire a deeper understanding of catalogue services. The specification encompasses three interrelated views that reflect different viewpoints on a catalogue service. Each viewpoint[2] focuses on different areas of concern:

- *Enterprise* – describes the general capabilities of the service in light of functional and non-functional requirements (for catalogue users and system testers);
- *Information* – defines the kinds of information handled by the catalogue and the policies to be enforced (for catalogue users, developers, and testers);
- *Computational* – specifies the public interfaces, allowable interactions, and protocol bindings (for developers and testers).

This document defines an application profile of an OGC Catalogue service; it is primarily based on the HTTP binding (the CSW part) described in Clause 11 of the OGC Catalogue Services Specification, version 2.0 (OGC 04-021r3). The profile constrains the usage of several base specifications and introduces some additional search, retrieval, and transaction capabilities.

The terms ‘catalogue’ and ‘registry’ are often used interchangeably, but the following distinction is made in this application profile: a registry is a specialized catalogue that exemplifies a formal registration process such as those described in ISO 19135 or ISO 11179-6. A registry is typically maintained by an authorized registration authority who assumes responsibility for complying with a set of policies and procedures for accessing and managing registry content. This profile does not stipulate any particular registration policies that must be enforced by a conforming implementation.

[2] The Reference Model of Open Distributed Processing (RM-ODP, ISO/IEC 10746) is the architectural framework adopted by the OGC and ISO/TC 211 for specifying software-intensive systems. In IEEE 1471 terminology the RM-ODP framework provides a set of library viewpoints.

OpenGIS® Catalogue Services — ebRIM (ISO/TS 15000-3) profile of CSW

1 Scope

A catalogue implementation that conforms to this profile can serve many purposes in a variety of application domains; it provides facilities for discovering and advertising a wide variety of information resources. While such resources are often labelled as "metadata", it is rarely possible to maintain an absolute distinction—what is deemed data in one context may be considered metadata in another. The catalogue information model is a general and flexible one that can be employed to handle many kinds of information resources—artifacts—including but not limited to: service offers, interface definitions, dataset descriptions, application schemas, and classification schemes. The service may be used to catalogue resources located in both local and remote repositories. Representations of these resources are exchanged using the standard HTTP/1.1 protocol.

2 Conformance

Conformance with this specification shall be checked using all the relevant tests specified by the Abstract Test Suite (ATS) in Annex A (normative). The framework, concepts, and methodology for testing, and the criteria to be achieved to claim conformance are specified in ISO 19105: Geographic information — Conformance and Testing.

In addition to satisfying the requirements stipulated in all normative clauses and Annex A, a catalogue implementation must also satisfy all relevant requirements in the following base specifications:

- OGC Catalogue Services 2.0, Clause 10 (OGC 04-021r3, with Corr. 1)
- OGC Web Services Common Specification 1.0 (OGC 05-008)
- OGC Filter Encoding Implementation Specification 1.1 (OGC 04-095)
- OASIS ebXML Registry Information Model, Version 3.0

3 Normative references

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

ISO 19105:2000, *Geographic information — Conformance and Testing*.

ISO/IEC 19106:2003, *Geographic Information – Profiles.*

ISO/IEC 19108:2002, *Geographic Information – Temporal schema*.

ISO/IEC 19119:2003, *Geographic Information – Services*

ISO/IEC 19125-1:2004, *Geographic Information – Simple Feature Access – Part 1: Common architecture*.

ISO/CD 19136, *Geographic Information – Geography Markup Language*.

IETF RFC 2388, *Returning values from forms: multipart/form-data*, Proposed Standard (August 1998), available [online]: <http://www.apps.ietf.org/rfc/rfc2388.html>.

IETF RFC 2616, *Hypertext Transfer Protocol -- HTTP/1.1*, Draft IETF Standard (June 1999), available [online]: <http://www.apps.ietf.org/rfc/rfc2616.html>.

OASIS ebRIM, *ebXML Registry Information Model Version 3.0*, OASIS Standard (May 2005), available [online]: <http://www.oasis-open.org/committees/download.php/13591/docs.oasis-open.orgregrepv3.0specsregrep-rim-3.0-os.pdf>.

OGC 04-095, *Filter Encoding Implementation Specification*, version 1.1.0 (3 May 2005), available [online]: <http://portal.opengeospatial.org/files/?artifact_id=8340>.

OGC 04-021r3, *OGC™ Catalogue Services Specification*, version 2.0. (May 2005, includes Technical Corrigendum 1).

OGC 05-008, *OGC Web Services Common Specification*, version 1.0 (May 2005).

In addition to this document, this specification includes several normative XML Schema files, which are available online in the OGC schema repository at this base URL <http://schemas.opengis.net/csw-ebrim/1.0.0>. These XML Schema files are also bundled with the present document. In the event of a discrepancy between the bundled and online versions of the XML Schema files, the online resources shall be considered authoritative.

4 Terms and definitions

For the purposes of this specification, the definitions specified in Clause 4 of the *OGC Web Services Common Specification* [OGC 05-008] shall apply. In addition, the following terms and definitions apply.

4.1
extension package
cohesive set of registry objects and related artifacts that extend the core information model to meet the needs of some application domain or community of practice.

NOTE Examples: a Geodesy package for CRS definitions; a Gazetteer package for feature registries.

4.2
extrinsic object
registry object that describes a repository item, the content of which is not constrained by the information model.

NOTE Examples include: an XML Schema, a data set description, a thumbnail image.

4.3
registration
assignment of an unambiguous identifier to an administered item in a way that makes the assignment available to interested parties. [ISO 11179-6]

4.4
registry object
administered item (an instance of the ebXML registry information model) published in a registry.

4.5
version
resource that contains a copy of a particular state of a version-controlled registry object or repository item.

4.6
version history
set containing all versions of a particular version-controlled resource.

5 Conventions

5.1 Abbreviated terms

Most of the abbreviated terms listed in Subclause 5.1 of the OWS Common Implementation Specification [OGC 05-008] apply to this document, plus the following abbreviated terms.

ATS	Abstract test suite
ebRIM	ebXML Registry Information Model
SUT	System Under Test
WSDL	Web Services Description Language

5.2 UML notation

Some of the diagrams that appear in this document are presented using the Unified Modeling (UML) notation. Subclause 5.2 of [OGC 05-008] provides some general guidance regarding the use of class diagrams.

5.3 Namespace prefix conventions

Table 1 lists the namespaces used in this document and the specifications in which they are defined. The prefixes are **not** normative and are merely chosen for convenience; they may appear in examples without being formally declared, and have no semantic significance. The namespaces to which the prefixes correspond are normative, however.

Table 1 — Namespace mappings

Prefix	Namespace URI	Specification
wrs	http://www.opengis.net/cat/wrs	CSW-ebRIM profile
rim	urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0	OASIS ebRIM 3.0
csw	http://www.opengis.net/cat/csw	CSW part (Clause 10) of OGC Catalogue Services 2.0
ows	http://www.opengeospatial.net/ows	OGC Common 1.0
ogc	http://www.opengis.net/ogc	OGC Filter 1.1
gml	http://www.opengis.net/gml	OGC GML 3.1.1
dc	http://purl.org/dc/elements/1.1/	Namespace Policy for the DCMI[a]
xlink	http://www.w3.org/1999/xlink	XML Linking Language (XLink) Version 1.0

a See <http://dublincore.org/documents/dcmi-namespace/>.

6 Overview

6.1 Essential capabilities

A service-oriented architecture must support some fundamental interactions: publishing resource descriptions so that they are accessible to prospective users (*publish*); discovering resources of interest according to some set of search criteria (*discover*); and then interacting with the resource provider to access the desired resources (*bind*). Within such an architecture a catalogue service plays the essential role of matchmaker by providing publication and search functionality, thereby enabling a requester to dynamically discover and communicate with a suitable resource provider without requiring the Requester to have advance knowledge about the Provider (Figure 1).

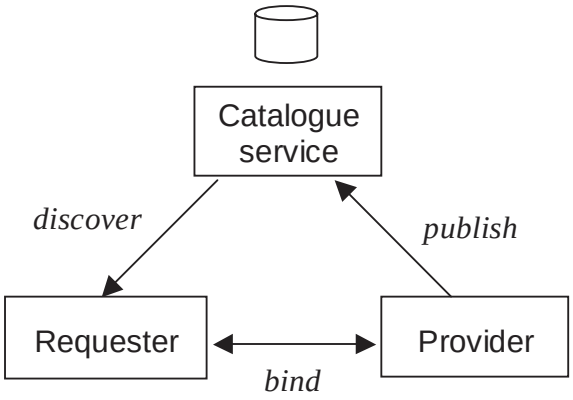

Figure 1 — Essential interactions in a service-oriented architecture

The essential purpose of a catalogue service is to enable a user to locate, access, and make use of resources in an open, distributed system by providing facilities for retrieving, storing, and managing many kinds of resource descriptions. The metadata repository managed by the catalogue can store a welter of resource descriptions that conform to any standard Internet media type, such as: XML schemas, thumbnail representations of remotely sensed images, audio annotations, specification documents, a simple map of a meteorological sensor network, and style sheets for generating detailed topographic maps. Furthermore, arbitrary relationships among catalogued items can be expressed by creating links between any two resource descriptions. For example, a service offer may be associated with descriptions of the data sets that can be acquired using the service.

6.2 ebXML registry information model (ebRIM)

The information model is based on the OASIS ebXML Registry Information Model (ebRIM), version 3.0. This logical model specifies how catalogue content is structured and interrelated; it constitutes a public schema for discovery and publication purposes. The UML representation shown in Figure 2 is intended to provide a condensed, informative overview of the information model—it is not normative. The OASIS standard defines a normative XML Schema for representing registry objects.

The information model is a general and flexible one with several extensibility points. A cohesive set of extensions that address the needs of a particular application domain or community of practice are declared as parts of an extension package. An extension package is a RegistryPackage that has some membership restrictions (Subclause 6.4).

An instance of rim:RegistryObject (or one of its subtypes) may be included in a CSW message in any context where a record representation is allowed; for example, as a child of the csw:SearchResults element in a GetRecordsResponse message. The ebRIM representations are provided if the value of the outputSchema attribute specified in the GetRecords request matches the namespace of the ebRIM schema. The ebRIM registry object representation must be used in all insert and update statements appearing within a csw:Transaction request.

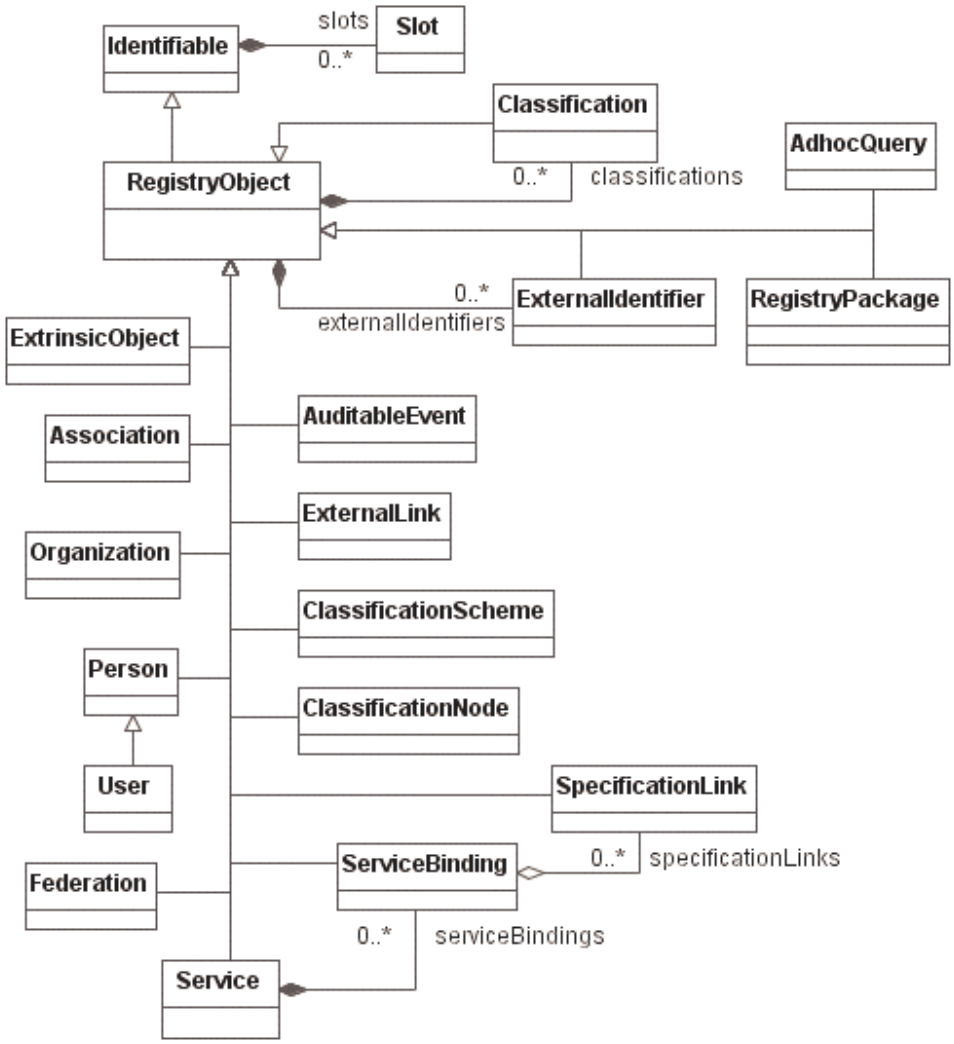

Figure 2 — High-level view of ebRIM 3.0 (attributes suppressed)

6.3 Repository items

The ebRIM defines an ExtrinsicObject type as a sort of proxy for an internally (or externally) managed repository item that conforms to some well-known content type, as indicated by the value of the rim:ExtrinsicObject/@mimeType attribute; this value must correspond to a registered MIME media type. The service capabilities document lists supported content types as values of the “mime-types” service property (see Subclause 8.1.4).

6.4 Extension packages

This application profile is intended to provide a flexible, general-purpose catalogue service that can be adapted to meet the needs of diverse communities of practice within the geospatial domain. The principal means of customizing the behaviour and content of

a catalogue service is by defining an extension package to exploit the extensibility points offered by ebRIM that enable it to be tailored for specific purposes; these extensibility points include:

- additional object types as specified by the value for the ExtrinsicObject.objectType attribute;
- new kinds of associations that link registry objects, as specified by the value of the Association.associationType attribute;
- additional classification schemes—or classification nodes that augment an existing scheme—for classifying registry content;
- stored queries that reflect common search patterns;
- additional slots to further characterize particular types of registry objects.

The **Basic** extension package is defined as part of this application profile—all implementations must support it (see Annex B). Additional packages may be defined by other parties as needed; these employ elements of ebRIM and the Basic package as suggested in Figure 3.

For example, a ‘Mapping’ package might include elements for working with the style descriptors and symbol collections used in map production. A ‘Geodesy’ package can include elements for defining coordinate reference systems and related components such as a datum and a prime meridian.

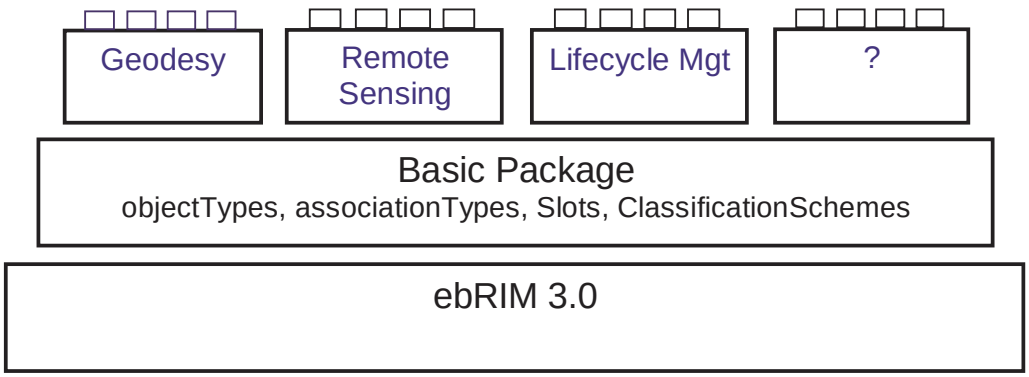

Figure 3 — Extension packages

An extension package is represented as a rim:RegistryPackage instance. Package members are registry objects that are subject to the following constraint: a member object may only be deleted if the package as a whole is deleted—this effectively treats an extension package as a composition.

6.5 Interfaces

6.5.1 OGCWebService

The `OGCWebService` interface is common to all OGC web service implementations. It provides the following operations:

a) **GetCapabilities** (required) – allows a client to retrieve service metadata that describe the computational and non-computational characteristics of the service.

6.5.2 Discovery

The `Discovery` interface provides the following operations:

a) **GetRecords** (required) – the principal operation used to search catalogue content and retrieve all or some members of the result set.

b) **DescribeRecord** (required) – allows a client to discover the information model(s) supported by the catalogue and to retrieve type definitions.

c) **GetRecordbyId** (required) – a simple means of retrieving one or more registry objects by identifier.

d) **GetDomain** (optional) – produces a description of the value domain of a given data element or request parameter, where the value domain may be enumerated or non-enumerated.

e) **GetRepositoryItem** (required) – requests the repository item for some extrinsic object.

6.5.3 Publication

The `Publication` interface provides the following operations:

a) **Harvest** (required) – enables a 'pull' style of publication whereby a resource is retrieved from some remote location (URL) and inserted into the catalogue.

b) **Transaction** (optional) – Allows a client to directly insert, update, or delete catalogue content.

Figure 4 is a UML diagram summarizing the service interfaces. Those operations that are not required to be implemented are shown as *abstract* operations. Each operation is described in more detail in subsequent clauses.

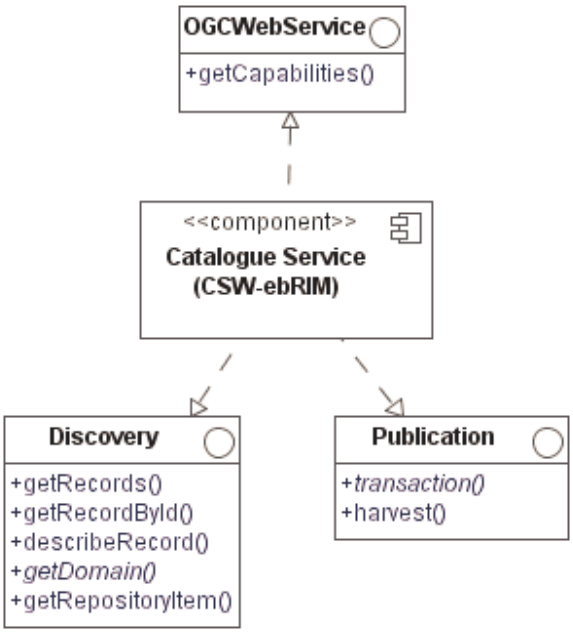

Figure 4 — Service interfaces (CSW-ebRIM)

6.6 Application schemas

This profile defines several normative XML Schemas, all of which reside in the "http://www.opengis.net/cat/wrs" namespace. The schemas are listed in Table 2; they are assigned the following resource identifiers:

- http://schemas.opengis.net/csw-ebrim/1.0.0/pkg-basic.xsd
- http://schemas.opengis.net/csw-ebrim/1.0.0/wrs-discovery.xsd
- http://schemas.opengis.net/csw-ebrim/1.0.0/wrs-capabilities.xsd
- http://schemas.opengis.net/csw-ebrim/1.0.0/wrs-publication.xsd

Table 2 — The CSW-ebRIM schemas

Schema name	Description
pkg-basic.xsd	Defines schema components for the Basic extension package.
wrs-discovery.xsd	Defines message elements for specialized search and retrieval operations.
wrs-capabilities.xsd	Defines additional elements used to describe profile-specific service capabilities.
wrs-publication.xsd	A simple composite schema for message elements pertaining to publication interactions.

The schemas are available online in the main OGC schema repository at this URL: <http://schemas.opengis.net/csw-ebrim/1.0.0/>. They are also distributed with this

document as a companion Zip archive. In the event of a discrepancy between the bundled and online versions of the schemas, the online files shall be considered authoritative.

Figure 5 is a UML package diagram illustrating the main schema dependencies (the common OWS schemas are omitted for clarity). The namespace prefixes map to namespaces as indicated in Table 1. Dependency relationships that cross namespace boundaries represent an *import* relationship; within a namespace they depict an *include* relationship.

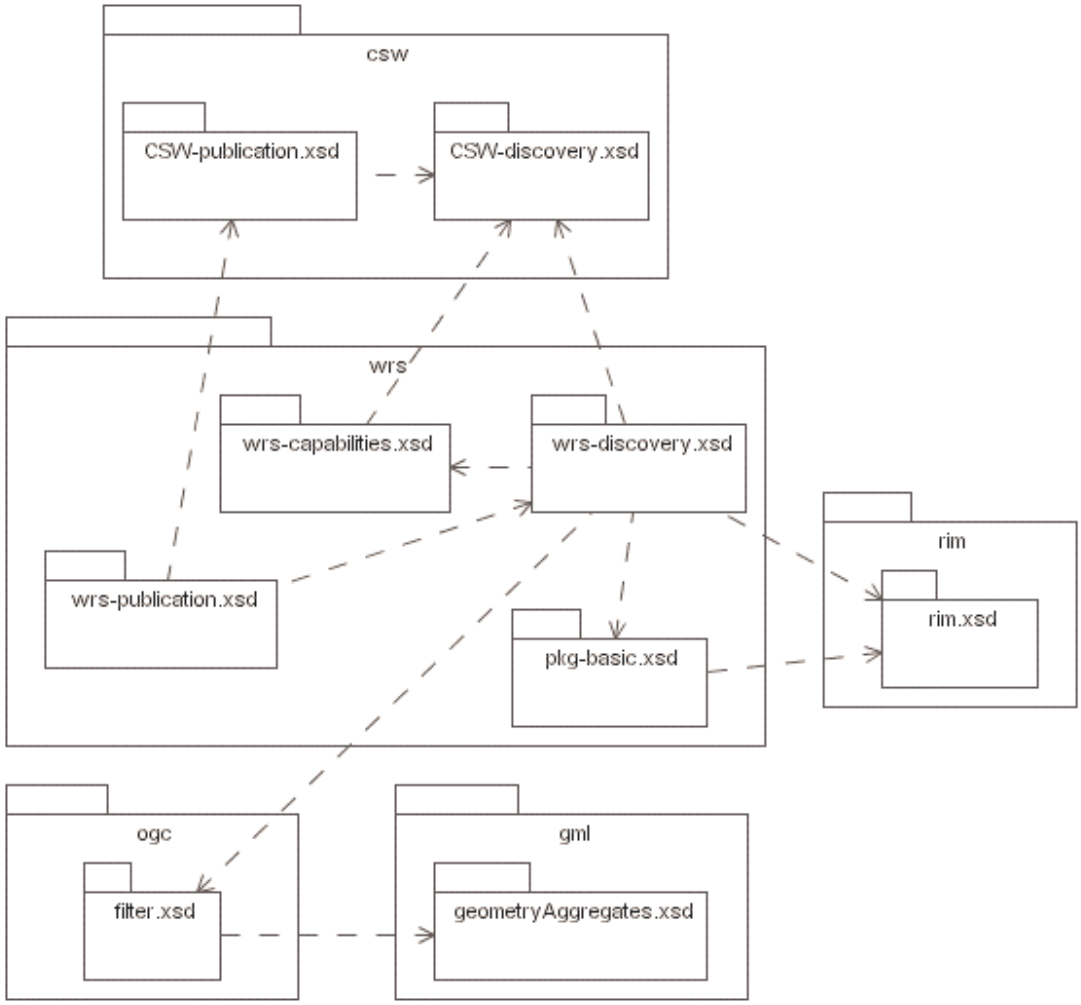

Figure 5 — Main schema dependencies

7 General capabilities

7.1 HTTP method bindings

The HTTP/1.1 specification [RFC 2616] defines eight methods for manipulating and retrieving representations of resources. Only the GET and POST methods are used in this application profile. All of the service operations defined in this application profile must be bound to HTTP methods as indicated in Table 3. Optional method bindings are indicated in parentheses. The encodings of the message payloads for all CSW requests are specified in Clause 10 of the OGC Catalogue Services 2.0 specification [OGC 04-021r3], as defined by the v2.0.1 schema set.

Table 3 — HTTP method bindings

Operation name[a]	HTTP method bindings[b]
GetCapabilities	GET (POST)
GetRecords	POST (GET)
GetRecordById	GET (POST)
DescribeRecord	POST (GET)
GetDomain	POST (GET)
GetRepositoryItem	GET
Transaction	POST
Harvest	POST
a Abstract operations are shown in *italics*. If not implemented, the response must indicate a status code of 501 (Not Implemented); the message body may also include an ows:ExceptionReport element. b Optional method bindings are enclosed in parentheses ().	

All bindings must be consistent with HTTP/1.1 semantics. For example, the GET method is used to retrieve whatever information—in the form of an entity—is identified by the Request-URI; it is deemed to be a “safe” method that does not give rise to any side effects such as modifying catalogue content.

7.2 Use of common message header fields

The HTTP specification defines a variety of message header fields that may be used to express client preferences or to supply metadata about the message body. Proper use of the Content-Type header is required by this profile; use of the other header fields appearing in Table 4 is strongly recommended. If any of these headers are included in a request message, they should be honored.

Table 4 — Common HTTP message headers

Header name	Purpose
Accept	Specifies acceptable MIME media types for the response[a].
Accept-Charset	May be used to indicate what character sets are acceptable

Header name	Purpose
	for the response[b]
Accept-Encoding	Indicates acceptable content codings for the response (e.g. gzip)
Accept-Language	Expresses a preferred natural language for the response
Content-Encoding	Identifies an encoding that has modified the media type of the entity-body (e.g. gzip).
Content-Type	Indicates the MIME media type of the enclosed entity-body
Last-Modified	Indicates the date and time at which the enclosed resource was last modified

a See the IANA media type registry at <http://www.iana.org/assignments/media-types/>.

b See the IANA character set registry at <http://www.iana.org/assignments/character-sets>.

7.3 Exception reports

Various faults or error conditions may arise while processing a request message. Several of these are predefined in this profile; they are included within a general service exception report that has <ows:ExceptionReport> as the document element. The report may include one or more <ows:Exception> elements that describe some error condition. The exception codes are qualified names where the prefix must be mapped to the "http://www.opengis.net/cat/wrs" namespace. The exception codes and the corresponding fault conditions are listed in Table 5. A generic exception code may be supplied if a more specific code does not apply.

Table 5 — Exception codes

Exception code	Fault condition
wrs:Sender	Intended for cases in which the message sender seems to have erred in some manner (this corresponds to an HTTP status code of 4xx).
wrs:Receiver	Intended for cases in which an unexpected condition prevented the service from fulfilling the request (this corresponds to an HTTP status code of 5xx).
wrs:InvalidRequest	The request message is either invalid or is not well-formed.
wrs:TransactionFailed	The requested transaction could not be completed.
wrs:NotImplemented	The (abstract) operation has not been implemented.
wrs:NotFound	The requested resource does not exist or could not be found.
wrs:NotSupported	A service option, feature, or capability is not supported.

Every operation may produce exceptions containing the wrs:InvalidRequest code if the request message is invalid in any way. If the Request-URI contains any unrecognized query parameters, these shall be ignored—an exception must not be generated.

7.4 Predefined property sets

The OGC Catalogue Services 2.0 specification distinguishes three abstract property sets—predefined views—that provide differing levels of detail about a catalogue item: brief, summary, and full. These abstract views are mapped to ebRIM registry objects as indicated in Table 6. The brief view represents a "minimal" registry object that is valid against the standard ebRIM schema. Note that the brief and summary views will generate <rim:RegistryObject> instances for any object type.

Table 6 — Registry object views

View name	ebRIM information items included
Brief	rim:RegistryObject/@id rim:RegistryObject/@lid rim:RegistryObject/@objectType rim:RegistryObject/@status rim:RegistryObject/rim:VersionInfo
Summary	*As for Brief view, plus:* rim:RegistryObject/rim:Slot rim:RegistryObject/rim:Name (in preferred languages) [a] rim:RegistryObject/rim:Description (in preferred languages) [a]
Full	Complete Information Set
a As specified by the value of the the Accept-Language request header field.	

7.5 Spatial references

Geospatial information resources often include spatial references of interest to users; such references encompass location identifiers such as place names or country subdivisions as well as geographic coordinates. Both kinds may be used to characterize registry objects using slots, where the slotType attribute specifies the type of reference provided.

The Basic extension package includes three slots that may be used to describe the spatial characteristics of a registry object. These derive from DCMI metadata terms and ISO 19136 (GML). The slot name is an absolute URI as indicated in Table 7. Simple slot values are contained as child elements of the standard rim:ValueList element. Complex values must have wrs:ValueList as the parent element, as this type defines a more flexible content model; whenever the wrs:ValueList element appears, the slotType attribute must also be present.

Table 7 — Spatial slots

Slot name	Slot type
http://purl.org/dc/elements/1.1/coverage	Provides a URI reference to an internal or external geocoding scheme (e.g. a thesaurus or classification scheme), or absent if using an uncontrolled vocabulary.

Slot name	Slot type
http://purl.org/dc/terms/spatial	A qualified type name (QName) that identifies a GML geometry type[a]
http://www.opengis.net/gml/Envelope	gml:EnvelopeType
a The following 7 basic geometry types are sufficient for most catalogue applications: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, MultiGeometry.	

Every geometry value must be associated with a coordinate reference system. No default reference system is identified in this profile, although the World Geodetic System 1984 (WGS 84) is widely used. Supported coordinate reference systems must be identified using the URN syntax documented in [OGC 05-010], and listed in the service capabilities document as values of the "http://www.opengis.net/cat/wrs/properties/spatial-ref-systems" property (see Subclause 8.1.4). If multiple geometry values are included for a given slot, they must represent the same geometry instance in different coordinate reference systems.

EXAMPLE 1 A Coverage slot providing country names from ISO 3166-1

```
<rim:Slot name="http://purl.org/dc/elements/1.1/coverage"
  slotType="urn:x-ogc:specification:csw-
ebrim:ClassificationScheme:ISO-3166-1:Countries">
  <rim:ValueList>
    <rim:Value>Canada (CA)</rim:Value>
    <rim:Value>United States (US)</rim:Value>
  </rim:ValueList>
</rim:Slot>
```

EXAMPLE 2 An Envelope slot indicating approximate spatial extent (WGS 84 coordinates, 2D)

```
<rim:Slot name="http://www.opengis.net/gml/Envelope"
  slotType="gml:EnvelopeType">
  <wrs:ValueList>
    <wrs:AnyValue>
      <gml:Envelope srsName="urn:x-ogc:def:crs:EPSG:6.7:4326"
                    srsDimension="2">
        <gml:lowerCorner>48.718 -124.07</gml:lowerCorner>
        <gml:upperCorner>49.833 -122.21</gml:upperCorner>
      </gml:Envelope>
    </wrs:AnyValue>
  </wrs:ValueList>
</rim:Slot>
```

7.6 Mapping registry objects to CSW records

Clause 10 of the OGC Catalogue Services 2.0 specification defines a simple record format that is common to all CSW-based catalogue services. A csw:Record instance is an XML document that contains a collection of Dublin Core metadata terms. The simple record types corresponding to "brief", "summary" and "full" views are all defined in the CSW schema, record.xsd.

Table 8 shows how selected CSW record properties are mapped to ebRIM information items. If no mapping is explicitly specified, then the property may be mapped to a Slot, where the name of the slot must match the absolute URI for the term in the DCMI specification.

Table 8 — Partial mapping of csw:Record elements to ebRIM

CSW property name	ebRIM information item(s)
dc:identifier[a]	rim:RegistryObject/@id rim:RegistryObject/rim:ExternalIdentifier/@value
dc:type	rim:RegistryObject/@objectType
dc:title	rim:RegistryObject/rim:Name
dc:description	rim:RegistryObject/rim:Description
dc:subject[b]	rim:RegistryObject/rim:Slot/rim:ValueList/rim:Value
dc:format	rim:ExtrinsicObject/@mimeType
ows:BoundingBox[c]	rim:RegistryObject/rim:Slot

a The first identifier (in document order) is taken as the principal identifier; remaining identifiers are considered to be external identifiers.

b This accommodates keywords from uncontrolled vocabularies; multiple values give rise to multiple properties. The corresponding slot name is "http://purl.org/dc/elements/1.1/subject". Note that formal classifications from ebRIM are not mapped.

c The corresponding slot name is "http://www.opengis.net/gml/Envelope".

7.7 Multilingual property values

Multilingual descriptors are accommodated through the use of the rim:InternationalString element that is the value of the RegistryObject.name and RegistryObject.description properties. This element is composed of a collection of rim:LocalizedString child elements that contain language-specific values for these properties.

Example Using multilingual descriptors

```
<rim:ExtrinsicObject
  id="${some-uri}"
  objectType="urn:x-ogc:specification:csw-ebrim:ObjectType:Dataset"
  xmlns:rim="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0"
  xmlns:xml="http://www.w3.org/XML/1998/namespace">

  <rim:Description>
    <rim:LocalizedString xml:lang="en"
      value="The National Air Photo Library (NAPL) has over six
      million aerial photographs." />
    <rim:LocalizedString xml:lang="fr"
      value="Nous conservons à la Photothèque nationale de l'air
      (PNA) plus de six millions de photographies aériennes." />
  </rim:Description>
</rim:ExtrinsicObject>
```


8 OGCWebService interface

8.1 GetCapabilities

8.1.1 Summary

The mandatory GetCapabilities operation allows a client to retrieve information about the service. The body of the response message contains a service description that advertises the basic operational and non-operational capabilities of the service; the description is structured into subsections that may be retrieved individually by name.

8.1.2 GetCapabilities request

The GetCapabilities operation is described in Clause 7 of OGC 05-008 (OWS Common 1.0) and Subclause 10.5 of OGC 04-021r3 (Catalogue 2.0). Note that all parameter names in the KVP-encoded request must be treated in a case insensitive manner, and the query component of the Request-URI in the GET request must be appropriately escaped. Support for the GET method is mandatory; the POST method is optional, using either of the allowed content encodings (application/xml or application/x-www-form-urlencoded).

The value of the *service* parameter shall be the following service type identifier: “urn:x-ogc:specification:csw-ebrim:Service:OGC-CSW:ebRIM”. When included within a query component in the Request-URI, the “:” character (COLON) must be percent-encoded as “%3A”, since it is data in this context, not a delimiter.

The *sections* parameter may be used to request a subset of the complete capabilities document; the value is a comma-separated list of section names. The allowed set of section names recognized by this profile are listed in Table 9. If this parameter is absent, the complete description must be returned; unrecognized section names are ignored.

Table 9 — Permissible section names

Section name	Content
ServiceIdentification	General information about the service (type, version, etc.).
ServiceProvider	Information about the organization providing the service.
OperationsMetadata	Summarizes the operational characteristics of the service.
Filter_Capabilities	Describes supported OGC filter operators
ServiceFeatures	Information about implemented features (see Subclause 8.1.4)
ServiceProperties	Information about general service properties (see Subclause 8.1.4)

8.1.3 GetCapabilities response

If the request is processed successfully, the body of the response message shall include an XML document where the document element has the following infoset properties:

- a [local name] of “Capabilities”.
- a [namespace name] of “http://www.opengis.net/cat/wrs”.

The document element **must** be valid against the following element declaration:

http://schemas.opengeospatial.net/csw-ebrim/1.0.0/wrs-capabilities.xsd#Capabilities

All of the top-level elements (sections) must be included if the *sections* request parameter was absent; otherwise only the specified top-level elements are included.

Several additional service metadata elements are introduced in this profile (Table 10).

Table 10 — Additional service metadata elements

Qualified name	Content
wrs:ServiceFeatures	A list of implemented features. See Subclause 8.1.4.
wrs:ServiceProperties	A list of general service properties. See Subclause 8.1.4.
wrs:WSDL-services	A reference to a W3C WSDL service description. See Subclause 8.1.5.

8.1.4 General service features and properties

A standard set of service features and properties are used to convey service capabilities that are specific to CSW-ebrim catalogues. A feature is essentially a boolean property indicating that a specific feature is available. Service properties may be multi-valued. All feature and property names are absolute URIs. Several general ones are defined in this specification, but providers may introduce new ones based on URIs they manage.

Table 11 lists the general WRS service features. The asterisk (*) denotes the prefix "http://www.opengis.net/cat/wrs/features", which is omitted from the table for convenience. An unimplemented feature will not be listed in the capabilities document. If some request depends on a feature that is not available, then a service exception must be returned with an exception code of "wrs:NotSupported".

Table 11 — General service features

Feature identifier	Description
*/deep-search	The service can execute searches against the content of repository items (the child mime-types property indicates which content types are searchable). See Subclause 9.1.7.
*/audit-trail	The service maintains an audit trail that tracks who changed what and when. See Subclause 10.2.2.5.
*/version-control	The service will treat instances of designated object types as versionable resources; the child object-types property indicates which types are versioned automatically. See Subclause 10.2.2.6.

Table 12 lists the general WRS service properties. The asterisk (*) denotes the prefix "http://www.opengis.net/cat/wrs/properties/", which is again omitted for convenience.

Table 12 — General service properties

Property identifier	Description
*/harvest-protocols	List of application protocols that the catalogue supports for harvesting remote content. The values are protocol names and versions appended with specification references in parentheses. Example: "HTTP/1.1 (http://www.apps.ietf.org/rfc/rfc2616)"
*/query-languages	List of supported query languages
*/mime-types	List of supported Internet media (MIME) types for repository items
*/extension-packages	List of installed extension packages
*/geometry-types	List of supported geometry types from ISO 19125-1 (GML 3 and WKT representations).
*/temporal-functions	List of supported functions that implement temporal relationships from ISO 19108 (see Subclause 9.1.10).
*/temporal-ref-systems	List of URI values identifying supported temporal reference systems. ISO 8601 is mandatory and the default if none is specified: "urn:x-ogc:def:trs:ISO-8601".
*/spatial-ref-systems	List of URI values identifying supported spatial reference systems (must conform to the URN syntax documented in OGC 05-010).
*/object-types	List of URI values that identify object types that will be automatically versioned; these must match *leaf* nodes in the canonical object types classification scheme.

8.1.5 Web Services Description Language (WSDL)

The Web Services Description Language (WSDL) is an XML language for describing the computational characteristics of a web service in terms of interfaces, protocol bindings, and service endpoints. WSDL 2.0 is currently a W3C Working Draft that defines a component model in terms of an abstract XML infoset.

A WSDL description may be used to complement the metadata provided in an OGC service capabilities document. A definition of the service interfaces is included in Annex C. These definitions are currently informative, pending release of the WSDL 2.0 specification as a W3C Candidate Recommendation.

The `<wrs:WSDL-services>` element is a simple link element that may be used to include a reference to a WSDL description containing service and binding elements. The value of the xlink:href attribute must be a resolvable URI that produces the WSDL document when it is the target of a GET request; the xlink:role attribute must indicate the relevant version of the WSDL specification (by namespace URI).

EXAMPLE Referencing a WSDL description in a capabilities document.

```
<wrs:WSDL-services xlink:type="simple"
    xlink:href="http://www.foo.net/wxs/wsdl"
    xlink:role="http://www.w3.org/2005/08/wsdl"
    xlink:title="Service endpoints and bindings (WSDL 2.0)" />
```


8.1.6 Exceptions

If an error condition arising while performing a GetCapabilities request, the service shall return an exception report as specified in Clause 8 of [OGC 05-008]. The allowed exception codes include those listed in Table 5 of Subclause 7.4.1 of [OGC 05-008], as well as `wrs:InvalidRequest` if applicable.

9 Discovery interface

9.1 GetRecords

9.1.1 Summary

The mandatory GetRecords operation is described in Subclause 10.8 of OGC 04-021r3 (Catalogue 2.0). This is the principal operation used to search catalogue content. Some or all of the registry objects in the result set that satisfy the search criteria may be "piggy-backed" in the response message. The POST method binding is mandatory, using either of the allowed content types; support for the application/xml content type is required in this case, however. The GET binding is optional.

9.1.2 GetRecords request

If the Content-Type of the request entity body is an XML content type (application/xml), then the document element must be the <csw:GetRecords> element, as defined in the following CSW schema:

http://schemas.opengeospatial.net/csw/2.0.1/CSW-discovery.xsd

A client may assign a request identifier (an absolute URI) for tracking or polling purposes, and several attributes may be specified to control retrieval behaviour. If the maxRecords attribute has the value "0" (zero) or the resultType attribute has the value "hits", then only a summary of the result set is included in the response. If the @resultType attribute has the value "validate", then the document element in the immediate response must be <csw:Acknowledgement> if the entity body is valid; if it is invalid, the response must include an exception report.

This profile does not currently allow for asynchronous processing. If the <csw:ResponseHandler> element is included it shall be ignored and the request will be processed in the normal synchronous fashion; the final response message will be returned directly to the user agent in the usual manner,

A csw:GetRecords element must include a query statement. The schema fragment below allows a choice between any element that can substitute for csw:AbstractQuery or any element from a namespace that is **not** "http://www.opengis.net/cat/csw". In practice only one of the following elements may appear in this context: a csw:Query element (see Subclause 9.1.4), a rim:AdhocQuery element (see Subclause 9.1.6), or a wrs:StoredQuery element (Subclause 9.1.8).

EXAMPLE Including query statements in a GetRecords request.

```
<xsd:choice>
  <xsd:element ref="csw:AbstractQuery"/>
  <xsd:any processContents="strict"
           namespace="##other" />
</xsd:choice>
```

If the <csw:DistributedSearch> element is present and the catalogue is able to perform a distributed search, then the catalogue will perform a local search and the request shall be forwarded to one or more affiliated catalogue services (see Subclause 9.1.9). Otherwise the <csw:DistributedSearch> element is ignored and only a local search is performed.

9.1.3 GetRecords response

If the request is processed successfully, the body of the response message shall include an XML document where the document element has the following infoset properties:

- a [local name] of “GetRecordsResponse”.
- a [namespace name] of “http://www.opengis.net/cat/csw”.

The RequestId element is included only if the client supplied a value in the original request. The search results may include a sequence of either <csw:Record> or <rim:RegistryObject> elements. In any case valid substitution elements may also be included, where these typically correspond to different views or instances of record subtypes.

The record representions must conform to the requested output schema. The value of the outputSchema attribute in the request restricts which elements may appear in the response (Table 13). If not specified, ebRIM representations are returned.

Table 13 — Allowable catalogue record representations

outputSchema	Record representations
http://www.opengis.net/cat/csw	csw:Record csw:SummaryRecord csw:BriefRecord
urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0	rim:RegistryObject *Any subtype of rim:RegistryObject*

9.1.4 csw:Query statements

The <csw:Query> element is documented in Subclause 10.8 of OGC 04-021r3. However, in order to adapt the generic query syntax to a complex information model such as ebRIM a few additional constraints are needed. Furthermore, the declaration of binding variables—or aliases—is often required to avoid ambiguity when specifying complex

queries that navigate associations by traversing multiple links between related registry objects.

The value of the Query/@typeNames attribute is a whitespace-separated list of object types that constitute the scope of the query. Each value in the list MUST be a qualified type name (QName). One or more variables may be bound to a type name as shown in the following example, where each variable ranges over instances of a given object type. Multiple variables may be bound to a single type; these are delimited by a comma.

EXAMPLE 1 Declaring binding variables srv1, srv2

```
<csw:Query typeNames="rim:Service=srv1,srv2 rim:ExtrinsicObject">
  <!-- query specification -->
</csw:Query>
```

EXAMPLE 2 Declaring a binding variable for a slot

```
<csw:Query typeNames="rim:Service rim:Slot=s1">
  <ogc:Filter>
    <ogc:And>
      <ogc:PropertyIsEqualTo>
        <ogc:PropertyName>rim:Service/$s1/@name</ogc:PropertyName>
        <ogc:Literal>http://purl.org/dc/terms/spatial</ogc:Literal>
      </ogc:PropertyIsEqualTo>
      <!-- additional predicates -->
    </ogc:And>
  </ogc:Filter>
</csw:Query>
```

The "$" (DOLLAR SIGN) character may be used to dereference a binding variable in the query filter.

EXAMPLE 3 Find data set descriptions associated with a service offer

```
<csw:Query typeNames='rim:ExtrinsicObject rim:Association=a1'>
  <csw:ElementSetName
    typeNames='rim:ExtrinsicObject'>full</ElementSetName>
  <csw:Constraint version='1.1.0'>
    <ogc:Filter>
      <ogc:And>
        <ogc:PropertyIsEqualTo>
          <ogc:PropertyName>$a1/@associationType</ogc:PropertyName>
          <ogc:Literal>OperatesOn</ogc:Literal>
        </ogc:PropertyIsEqualTo>
        <ogc:PropertyIsEqualTo>
          <ogc:PropertyName>$a1/@sourceObject</ogc:PropertyName>
          <ogc:Literal>
          urn:uuid:86084c6a-292f-4687-bf52-aef49b5ff2d6
          </ogc:Literal>
        </ogc:PropertyIsEqualTo>
        <ogc:PropertyIsEqualTo>
          <ogc:PropertyName>rim:ExtrinsicObject/@id</ogc:PropertyName>
          <ogc:PropertyName>$a1/@targetObject</ogc:PropertyName>
        </ogc:PropertyIsEqualTo>
        <ogc:PropertyIsEqualTo>
          <ogc:PropertyName>
```



```
            rim:ExtrinsicObject/@objectType
          </ogc:PropertyName>
          <ogc:Literal>
            urn:x-ogc:specification:csw-ebrim:objectType:Dataset
          </ogc:Literal>
        </ogc:PropertyIsEqualTo>
      </ogc:And>
    </ogc:Filter>
  </csw:Constraint>
</csw:Query>
```

9.1.5 Spatial operators

An OGC filter expression may contain a spatial operator that specifies a query against the spatial characteristics of a registry object. The OGC filter grammar [OGC 04-095] defines a number of operators that evaluate various spatial relationships documented in ISO 19125-1; these are boolean methods that are used to test for the existence of a specified topological spatial relationship between two geometric objects.

The CSW part of the OGC Catalogue Services 2.0 specification only requires support for the BBOX predicate—this is a convenience operator that is equivalent to **Not** Disjoint or Intersects relationship. This profile does not mandate support for any additional spatial operators, but all supported operators must be listed in the service capabilities document within the Filter_Capabilities section. If any filter expression contains an unsupported operator the service must return an exception with code wrs:NotSuppported.

9.1.6 rim:AdhocQuery statements

The rim:AdhocQuery element defined in ebRIM 3.0 permits the use of alternative query languages such as XPath and XQuery. The child <rim:QueryExpression> element encapsulates a query expressed in a specified query language. In the context of a GetRecords request the query expression is required; if it is missing, the catalogue must raise an exception with code wrs:InvalidRequest. The query language should correspond to one supported by the service, as advertised by the query-languages service property in the capabilities document; if the query language is unsupported, the catalogue must raise an exception with code wrs:NotSupported.

9.1.7 XPath predicates

The <wrs:XPathPredicate> element encapsulates an XPath expression to be evaluated against a repository item that conforms to some XML content type, thereby enabling a kind of “deep search” facility for searching extrinsic content not otherwise visible through ebRIM. This element may appear within an <ogc:Filter> expression wherever a comparison operator is allowed. All relevant namespace bindings used in the expression must be declared on the XPath predicate.

Example Definition of XPathPredicateType (from wrs-discovery.xsd).

```
<xsd:complexType name="XPathPredicateType" id="XPathPredicateType">
  <xsd:complexContent mixed="true">
```



```
    <xsd:extension base="ogc:ComparisonOpsType">
      <xsd:attribute name="typeName" type="xsd:string"
                     use="required" />
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The typeName attribute specifies the extrinsic object type to query; this may imply a set of repository items of a given kind, subject to any additional constraints specified in the query filter. For a given repository item, the result is converted to a boolean value as described in section 2.4 of the XPath 1.0 specification. Such predicates only apply to extrinsic objects for which the mimeType attribute indicates an XML media type and the isOpaque property has the value “false”.

If the catalogue service does not support the deep-search feature for XML content types, then an exception must be returned with the code wrs:NotSupported.

9.1.8 Stored queries

9.1.8.1 Invoking stored queries

Extension packages may define various stored queries as a convenience; these are essentially parameterized, named queries. There are two mechanisms for invoking a stored query, using the GET and POST methods. The supported method bindings must be described in the stored query definition (rim:AdhocQuery/rim:Description). The response must have either csw:GetRecordsResponse or rim:RegistryObjectList (if invoked using the GET method) as the document element. A service provider may restrict who can read or execute a given stored query.

Using the POST method with a Content-Type of “application/xml”, a <wrs:StoredQuery> element (an instance of the rim:IdentifiableType, as declared in pkg-basic.xsd) may appear in a GetRecords request instead of a <csw:Query> element. The id attribute specifies the stored query to execute; any parameters are passed in as child Slot elements. Only stored queries that produce a list of matching registry objects may be run in this context.

A KVP-style syntax may be used to invoke a stored query using the GET method (or POST method with Content-Type set to “application/x-www-form-urlencoded”). The value of the final path segment (shown as /query-name in the example below) is a query reference; the stored query is referenced by name. Additional parameters are appended as shown in the following example.

Example General syntax for invoking a stored query using the GET method. The query name is appended to the path component of the URL.

```
http://host:port/path/query-name?param1=v1&param2=v2a,v2b
```

If there is no matching stored query, the response message must indicate a status code of 404 (Not Found); the body may also include an exception report with code wrs:NotFound. If any required query parameters are missing, then an exception with code wrs:InvalidRequest must be returned.

9.1.8.2 Defining stored queries

A stored query is defined by a rim:AdhocQuery instance that may be managed like any other registry object. The query expression must conform to a supported query language. That is, the value of the rim:QueryExpression/@queryLanguage attribute must refer to a node in the canonical query language scheme (which may be extended as needed to accommodate additional query languages).

Formal query parameters are included as slots in the rim:AdhocQuery instance; parameter bindings in the query specification are indicated using $*parameter-name* expressions. Within a definition, the slotType attribute should specify the appropriate value space for actual parameters. The rim:ValueList element must be empty, unless a default value is declared.

9.1.9 Distributed search

The CSW part of the OGC Catalogue Services 2.0 specification does allow for distributed searching whereby queries are forwarded to affiliated catalogue services, effectively enlarging the search space. However this profile does not currently specify such a protocol; a service provider may implement a distributed search facility and advertise its availability as a provider-specific feature in the service capabilities description.

9.1.10 Temporal queries

Catalogue content will often have temporal attributes such as the date of last modification or the period over which a data set was collected. ISO 19108 distinguishes 13 temporal relationships that can be realized using boolean functions as part of a temporal constraint in a query filter. Available functions are listed in the service capabilities document as values of the “http://www.opengis.net/cat/wrs/properties/temporal-functions” property. Subclause 5.2.3.5 of ISO 19108 describes the conditions that must be satisfied by each relationship.

EXAMPLE A temporal constraint for finding registry objects modified since 1 Aug 2005.

```
<ogc:PropertyIsEqualTo>
  <ogc:Literal>TRUE</ogc:Literal>
  <ogc:Function name="During">
    <ogc:PropertyName>rim:AuditableEvent/@timestamp</ogc:PropertyName>
    <ogc:Literal>
      <gml:TimePeriod>
        <gml:beginPosition>2005-08-01T00:00-08:00</gml:beginPosition>
        <gml:endPosition indeterminatePosition="now" />
      </gml:TimePeriod>
    </ogc:Literal>
  </ogc:Function>
</ogc:PropertyIsEqualTo>
```

The value of the top-level ogc:Literal element must be a boolean value: “TRUE” or “FALSE” (case insensitive). The name of the function must correspond to one of the values appearing in Figure 4 of ISO 19108: Before, After, Begins, Ends, During, Equals,

Contains, Overlaps, Meets, OverlappedBy, MetBy, BegunBy, EndedBy. The child PropertyName element must refer to some temporal property. The literal temporal value must be a primitive GML temporal element: gml:TimeInstant or gml:TimePeriod.

Following ISO 19108, the primary—and default—temporal reference system is the Gregorian calendar with UTC (Coordinated Universal Time). The identifier for this reference system is “urn:x-ogc:def:trs:ISO-8601”.

9.2 GetRecordById

9.2.1 Summary

The GetRecordById operation provides a simple means of retrieving one or more records by identifier; the identifier may be that of some registry object (rim:RegistryObject/@id) or an external identifier (rim:ExternalIdentifier/@value) assigned to a registry object. Support for the GET method binding is mandatory. Implementing the POST method is optional, in which case the “application/xml” content type must be understood.

9.2.2 GetRecordById request

The GetRecordById operation is described in Subclause 10.9 of OGC 04-021r3 (OGC Catalogue Services 2.0). The XML representation of the entity body, if present, must conform to the csw:GetRecordById element declaration (see CSW-discovery.xsd). All reserved characters (e.g., general delimeters) appearing in identifier values must be suitably percent-encoded in the KVP representation when using the GET method.

The value of an Id message item (using the GET or POST methods) identifies a registry object either directly or by an external identifier that corresponds to a child rim:ExternalIdentifier element.

9.2.3 GetRecordbyId response

If the request is processed successfully, the body of the response message shall include an XML document where the document element has the following infoset properties:

- a [local name] of “GetRecordByIdResponse”.
- a [namespace name] of “http://www.opengis.net/cat/csw”.

The child elements must be registry object representations (i.e. rim:RegistryObject or some valid substitution element) corresponding to the requested property set. If a match for an external identifier is found, then the parent registry object is included. If there are no matching records, an empty response is returned

9.2.4 Exceptions

If the request is deemed invalid for any reason (e.g. missing a required element), then the service must return an ows:ExceptionReport containing a service exception with the code wrs:InvalidRequest.

9.3 DescribeRecord

9.3.1 Summary

The DescribeRecord operation allows a client to discover the information model(s) supported by the catalogue and to retrieve record type definitions. Support for the POST method binding is mandatory using either of the allowed content types, although the "application/xml" content type must be understood in this case. The GET binding is optional.

9.3.2 DescribeRecord request

The DescribeRecord operation is described in Subclause 10.6 of OGC 04-021r3 (OGC Catalogue Services 2.0). The XML representation of the entity body, if present, must conform to the csw:DescribeRecord element declaration (see CSW-discovery.xsd). The TypeName elements, if present, identify the model elements for which type definitions are requested. The targetNamespace attribute of the csw:TypeName element must be specified; it may have the value "##any" to indicate any namespace.

The only schema language currently supported by this profile is W3C XML Schema. The corresponding value of the schemaLanguage attribute is given by the following URI: "http://www.w3.org/2001/XMLSchema".

The outputFormat attribute may be used to specify an alternative MIME media type for the response. In most cases this involves transforming the raw schema to a more human readable representation. Such a capability is optional.

9.3.3 DescribeRecord response

If the request is processed successfully, the body of the response message shall include an XML document where the document element has the following infoset properties:

- a [local name] of "DescribeRecordResponse".
- a [namespace name] of "http://www.opengis.net/cat/csw"

If no TypeName elements were provided in the request, then all of the schemas defining the information model must be included within csw:SchemaComponent elements. If there are no matching schema components, the document element must be empty.

The content of a csw:SchemaComponent element may be a complete schema or a fragment of one. If it is a fragment, the parentSchema attribute must reference the source schema (by identifier).

9.3.4 Exceptions

If the request is deemed invalid for any reason (e.g. missing a required element), then the service must return an ows:ExceptionReport containing a service exception with the code wrs:InvalidRequest.

9.4 GetDomain

9.4.1 Summary

The optional GetDomain operation produces a description of the value domain of a given data element or request parameter, where the value domain is the set of actual or *permissible* values. The value domain may be enumerated or non-enumerated. One use of this operation is to discover 'active' terms in a taxonomy that are currently used to classify registry objects.

The actual type of the data element is always returned, even if additional information about the value space is available. Support for the POST method binding is mandatory using either of the allowed content types, although the "application/xml" content type must be understood in this case. The GET binding is optional.

9.4.2 GetDomain request

The GetDomain operation is described in Subclause 10.7 of OGC 04-021r3 (OGC Catalogue Services 2.0). The XML representation of the entity body, if present, must conform to the csw:GetDomain element declaration (see CSW-discovery.xsd). The value of the csw:PropertyName must be an XPath location path that references some infoset item. Many of the ebRIM properties can be accessed in this manner, and those that refer to nodes in classification schemes can be usefully queried in this manner.

As a special case, a request for a list of catalogued object types must identify only those types of registry objects that actually exist in the catalogue. That is, specifying a PropertyName value of "rim:RegistryObject/@objectType" will yield a list of object types that currently populate the catalogue, rather than a reference to the conceptual scheme that identifies all supported object types. In general, all properties that refer to canonical classification nodes should be handled in this manner.

EXAMPLE Request a listing of all catalogued object types.

```
<csw:GetDomain>
  <csw:PropertyName>
    rim:RegistryObject/@objectType
  </csw:PropertyName>
</csw:GetDomain>
```

9.4.3 GetDomain response

If the request is processed successfully, the body of the response message shall include an XML document where the document element has the following infoset properties:

- a [local name] of "GetDomainResponse".
- a [namespace name] of "http://www.opengis.net/cat/csw"

The csw:DomainValues element must be present. In general the service is not required to report the 'active' subset of the value domain, especially for cases where this may be

computationally expensive or is otherwise infeasible to determine. However, a qualified type name must always be returned.

9.4.4 Exceptions

If this operation has not been implemented, the body of the response must include an ows:ExceptionReport element containing a service exception with the code wrs:NotImplemented. If the request specifies an unknown information item, the response must include an exception with the code wrs:NotFound.

9.5 GetRepositoryItem

9.5.1 Summary

The GetRepositoryItem operation is used to retrieve the repository item corresponding to some extrinsic object. If available, the item is included in the body of the response; it must be an instance of a MIME media type, as indicated by the value of the Content-Type header field.

An extrinsic object may also be used to catalogue an external repository item that is managed by another party. In this case, the ExtrinsicObject must be associated (using the "RepositoryItemFor" association) with an ExternalLink that specifies an absolute URL for retrieving the item.

9.5.2 GetRepositoryItem request

The request is bound only to the GET method. All reserved characters appearing in parameter values must be suitably percent-encoded. The request parameters are listed in Table 14.

Table 14 — GetRepositoryItem request parameters

Name	Definition	Use
service	Service type identifier (fixed value: "urn:x-ogc:specification:csw-ebrim:Service:OGC-CSW:ebRIM")	Mandatory
request	Operation to invoke (fixed value: "GetRepositoryItem")	Mandatory
id	Absolute URI that refers to some extrinsic object.	Mandatory

9.5.3 GetRepositoryItem response

If the request is processed successfully and a repository item is accessible, the body of the response message shall include the repository item as a MIME entity. If any additional encodings have been applied to the resource (e.g., compression using gzip), these must be specified by the Content-Encoding header field.

In some cases the resource may reside in an external repository maintained by another party. In this case, the catalogue shall redirect the client using the standard HTTP

redirection mechanism (i.e., status code 303, “See Other”) and set the value of the `Location` header field according to the value of the ExternalLink/@externalURI attribute.

9.5.4 Exceptions

If the request is deemed invalid for any reason (e.g. missing identifier), then the service must return an ows:ExceptionReport containing a service exception with the code wrs:InvalidRequest. If the supplied identifier does not match any registry object or if a repository item cannot be located, the response must include an exception with the code wrs:NotFound.

10 Publication interface

10.1 Harvest

10.1.1 Summary

The Harvest operation is described in subclause 10.12 of OGC 04-021r3 (Catalogue 2.0). It allows a user to request that the catalogue attempt to harvest a repository item from a specified network location, thereby realizing a 'pull' model for publishing registry content. If the catalogue successfully retrieves the resource and successfully processes it, then one or more corresponding registry objects are created or updated. Brief representations of all modified records are returned to the client when processing is complete.

10.1.2 Harvest request

The request is only bound to the HTTP POST method. The XML representation of the entity body must conform to the csw:Harvest element declaration (see CSW-publication.xsd).

The csw:Source element specifies a URL from which the resource may be retrieved. The scheme component should correspond to a protocol supported by the catalogue; support for the “http” scheme is required by all conforming implementations, and “HTTP/1.1” must be listed in the capabilities document as a value of the “harvest-protocols” system property.

If specified, the csw:ResourceType element must indicate the object type of the corresponding extrinsic object. It may be possible for the catalogue to deduce this from the content of the resource (for example, a data set description that conforms to the ISO 19139 schemas). The value should correspond to a type supported by the catalogue, as identified in the objectType classification scheme.

10.1.3 Harvest response

If the request is processed successfully, the body of the response message shall include an XML document where the document element has the following infoset properties:

- a [local name] of “HarvestResponse”.
- a [namespace name] of “http://www.opengis.net/cat/csw”

The document element must include a csw:TransactionResponse element that contains the csw:InsertResults child element; this element must list all registry objects that were created as a result of the harvesting operation.

10.1.4 Exceptions

If the resource cannot be retrieved from the source URL, an exception with code wrs:NotFound must be included in an ogc:ExceptionReport. If the resource format is not supported by the catalogue or the object type is not recognized, an exception with code wrs:NotSupported must be returned. In the event that the transaction cannot be completed for any reason, an exception with code wrs:TransactionFailed must be returned.

10.2 Transaction

10.2.1 Summary

The Transaction operation is described in Subclause 10.11 of OGC 04-021r3 (Catalogue 2.0, Corr. 1). It allows users to insert, update, or delete registry objects. A transaction request constitutes an atomic unit of work: all subsidiary commands must be successfully executed or the entire transaction fails; this ensures the integrity of catalogue content, especially when a set of interrelated registry objects are being submitted.

Typically transaction requests are subject to some kind of access control such that only authorized users may perform such actions. While ebRIM mandates the use of XACML (the OASIS eXtensible Access Control Markup Language) to specify access control policies, this profile does not stipulate any particular security mechanisms. Service providers are free to define and enforce access control policies at several levels of granularity (e.g. service, registry object, repository item).

10.2.2 Transaction request

10.2.2.1 General behaviour

The transaction request is bound to the HTTP POST method. The XML representation of the entity body must conform to the csw:Transaction element declaration (see CSW-publication.xsd).

A registry object representation may be included in a transaction request wherever the xsd:any wildcard element appears in the definitions of the insert and update statements. No other elements may appear in this context, including instances of any of the common CSW record types. Transactions apply to elements of the core information model.

This profile does not currently allow for managing the lifecycle of registry content. No state transitions are defined, and in the absence of any kind of lifecycle management

facility the status of all registry objects is fixed at “Submitted”. If a catalogue provides such a facility it must comply with the ebRIM 3.0 specification.

10.2.2.2 Insert statements

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 csw: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.

10.2.2.3 Update statements

When specifying a partial update that modifies selected properties of one or more registry objects, the value of the RecordProperty/Name element must be an XPath location path that identifies the property to be updated. The RecordProperty/Value element is of type xsd:anyType, but when updating registry objects typically only simple values are provided; structured content may be included when updating complex slot values.

If a registry object is included in the context of an update statement, this will either replace the existing object or create a new version if it is a versioned object (see

Subclause 10.2.2.6). For non-versioned registry objects a missing information item is interpreted as a request to remove it or reset it to the default value (if defined).

10.2.2.4 Delete statements

The deletion of registry objects is subject to a some basic rules regarding composite references stipulated in section 2.5.2 of the ebRIM 3.0 specification. When a given registry object is deleted, this deletion is cascaded to certain types of component registry objects: rim:ExternalIdentifier, rim:Classification, rim:ServiceBinding, and rim:SpecificationLink.

A registry object can be deleted only if there are no references to that object. That is, the object must not be referenced from any other rim:Association or rim:ExternalLink object; such links are not subject to a cascading delete.

Whenever an extrinsic object is deleted, the corresponding repository item is also deleted (if it exists).

10.2.2.5 Maintaining an audit trail

A catalogue that implements this profile may maintain an audit trail for each registry object; this is indicated by adding the “audit-trail” feature to the capabilities document. An audit trail includes all of the events that have changed the state of the object (e.g., create, update, or delete requests); it consists of a sequence of AuditableEvent objects, as documented in the ebRIM specification. A user may access the audit trail by submitting a GetRecords request, but typically viewing the audit trail is subject to some level of access control (e.g. only the owner of a resource may view the audit trial).

10.2.2.6 Basic versioning

A catalogue that implements this profile may provide a basic form of linear versioning for specific types of registry objects, as indicated by adding the “version-control” feature to the capabilities document; the child object-types property identifies which types are versioned automatically. Since the rim:VersionInfo element is required when marshalling a registry object (see ebRIM, sec. 2.5.1), the versionName attribute must have the value “UNVERSIONED” for all unversioned registry objects. By default, only the latest version shall be included in a result set.

When a registry object or repository item is updated, a new version is automatically created and added to the version history. The version history is the set of all versions of a particular version-controlled resource—it includes all registry objects that have the same value for the lid attribute. Each version in the set must have a unique id value, which must be a valid within some formal or experimental URN namespace [RFC 3406].

The version naming scheme and the maximum size or age of the version history is left to the discretion of the service provider. The naming scheme should have a simple lexicographic ordering. The rim:VersionInfo/@comment attribute may be set by the client in an update request.

10.2.2.7 Modifying slots

Inserting, updating, or deleting a slot is always an update against some registry object. If a slot of the same name already exists, the slot values are replaced. A slot is deleted by providing an empty rim:ValueList element; that is, emptying a slot effectively removes it.

When modifying slots in such a fine-grained manner, the update statement must include a rim:Identifiable element in order to identify the parent registry object; it may contain one or more slots to be added, replaced, or removed.

EXAMPLE Adding and removing a slot.

```
<csw:Update>
  <rim:Identifiable
    id="urn:uuid:c8409773-177f-4266-bac9-fa98f73b5664">
    <rim:Slot name="http://purl.org/dc/elements/1.1/rights"
              slotType="xsd:string">
      <rim:ValueList>
         <rim:Value>Copyright 2005 Foo, Inc.</rim:Value>
      </rim:ValueList>
    </rim:Slot>
    <rim:Slot name="http://purl.org/dc/elements/1.1/coverage">
      <rim:ValueList />
    </rim:Slot>
  </rim:Identifiable>
</csw:Update>
```

10.2.3 Transaction response

If the request is processed successfully, the body of the response message shall include an XML document where the document element has the following infoset properties:

- a [local name] of “TransactionResponse”
- a [namespace name] of “http://www.opengis.net/cat/csw”

10.2.4 Exceptions

In the event that the transaction fails for any reason, the response message body must have ows:ExceptionReport as the document element. It shall contain an exception with code wrs:TransactionFailed.

Annex A
(normative)

Abstract test suite

A.1 Test module for general capabilities

A.1.1 General capabilities

b) Test purpose: Confirm that the SUT satisfies conformance requirements that generally apply to all service interactions.

c) Test method: Falsification testing of HTTP response.

d) Reference: OGC 05-025r1

e) Test type: Capability

A.1.2 Test case for Content-Type header

a) Test case identifier:
urn:x-ogc:specification:csw-ebrim:atc:general:ContentTypeHeaderTest

b) Test purpose(s): The Content-Type message header must correctly identify the media type of the entity-body in the response (if present).

c) Test method: Check the value of the Content-Type header. Pass if it correctly identifies the media type of the entity-body. Fail otherwise.

d) Reference: OGC 05-025r1, Subclause 7.2; IETF RFC 2616, 7.2.1; IETF RFC 3023.

e) Test type: Basic

A.1.3 Test case for valid XML entity body in response message

a) Test case identifier:
urn:x-ogc:specification:csw-ebrim:atc:general:ValidXmlRspTest

b) Test purpose(s): The body of a response message that specifies an XML content type shall be schema valid.

c) Test method: For all response messages that specify a Content-Type header value corresponding to an XML media type (per RFC 3023), validate the document element in the entity body against its type definition. Pass if validation succeeds. Fail otherwise.

d) Reference: OGC 05-025r1, Clause 8.3.

e) Test type: Basic

A.2 Test module for Discovery operations

A.2.1 Discovery operations

a) Test purpose: Confirm that the SUT satisfies conformance requirements that apply to operations provided by the Discovery interface.

b) Test method: Falsification testing of HTTP response.

c) Reference: OGC 05-025r1, Clause 9.

d) Test type: Capability

A.2.2 Test case for validation of GetRecords request

a) Test case identifier:
urn:x-ogc:specification:csw-ebrim:atc:discovery:ValidateGetRecordsReqTest

b) Test purpose(s): A service shall validate a GetRecords request if the csw:GetRecords/@resultType attribute has the value “validate”.

c) Test method: Check that the entity body in the request is *at least* schema valid; additional constraints may also apply (e.g., type name values must identify known types). Pass if **all** of the following are true: (a) the request is valid and the document element in the response is csw:Acknowledgement; (b) the request is invalid and the document element in the response is ows:ExceptionReport. Fail otherwise.

d) Reference: OGC 05-025r1, Subclause 9.1.2; IETF RFC 2616, 7.2.1; IETF RFC 3023.

e) Test type: Basic

Annex B
(normative)

The Basic extension package

B.1 Introduction

The Basic extension package introduces artifacts of general utility in the geomatics domain; these constitute a rather thin customization layer over ebRIM. All conforming implementations must deploy this package, the members of which are summarized in the following subclauses. The assigned package identifier complies with the 'ogc' URN scheme: "urn:x-ogc:specification:csw-ebrim:package:Basic".

The following canonical ebRIM classification schemes are required by this package:

- urn:oasis:names:tc:ebxml-regrep:classificationScheme:AssociationType
- urn:oasis:names:tc:ebxml-regrep:classificationScheme:EventType
- urn:oasis:names:tc:ebxml-regrep:classificationScheme:NodeType
- urn:oasis:names:tc:ebxml-regrep:classificationScheme:ObjectType
- urn:oasis:names:tc:ebxml-regrep:classificationScheme:QueryLanguage
- urn:oasis:names:tc:ebxml-regrep:classificationScheme:StatusType

B.2 Classification nodes

This package adds the extrinsic object types listed in Table B.1. These nodes extend the canonical ebRIM object type scheme. Each object type is assigned an identifier based on the 'ogc' URN scheme; the asterisk in the type identifier denotes the following string, which has been omitted in the table for convenience:

`urn:x-ogc:specification:csw-ebrim:ObjectType`

Table B.1 — Extrinsic object types included in the Basic package

Object type ID	Description
*:ServiceProfile	Describes what the service does: its features and capabilities (from the OWL-S service ontology). Example: an OGC capabilities document
*:ServiceModel	Describes how the service works, including its essential computational characteristics and behaviours (from the OWL-S service ontology). Example: a WSDL interface description
*:ServiceGrounding	Describes how to access the service: the communications protocols and network endpoints (from the OWL-S service ontology). Example: a WSDL service description

Object type ID	Description
*:Dataset	Description of a geographic data set (from ISO 19115).
*:Schema	A formal description of a conceptual model, expressed using a textual or graphical schema language (e.g., UML, XMI, XML Schema, RELAX NG, ASN.1).
*:Stylesheet	A set of rules for presenting or styling some information resource, typically expressed using a style language (e.g., XSLT, CSS).
*:Document	Documentation of any kind (e.g. specifications, manuals, reports, FAQs).
*:Annotation	Commentary intended to interpret, explain, or clarify some other resource or part thereof.
*:Image	A symbolic visual resource other than text (e.g., diagrams, photographs, drawings, maps, animations).
*:Rights	Information about the rights held in and over the resource. Typically, a Rights object embodies a rights management statement that stipulates conditions of use or distribution. Example: an ODRL or MPEG-21/REL statement

The association types listed in Table B.2 extend the canonical ebRIM association type scheme. For each classification node appearing in Table B.2 there is also a *meta-association* that specifies the types of the source and target objects. The object type identifier has the following form:

```
urn:x-ogc:specification:csw-ebrim:AssociationType
```

Table B.2 — Association types included in the Basic package

Association type ID	Description
*:OperatesOn	Associates a Service offer with a description of the data that the service operates on as input or output (from ISO 19119). **Source object type:** urn:oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject:Service **Target object type:** urn:x-ogc:specification:csw-ebrim:ObjectType:Dataset
*:Presents	Associates a Service offer with a description of what the service does (from the OWL-S 1.1 service ontology). **Source object type:** urn:oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject:Service **Target object type:** urn:x-ogc:specification:csw-ebrim:ObjectType:ServiceProfile
*:Supports	Associates a Service offer with a description of how an agent can access the service using some communication protocol (from the OWL-S 1.1 service ontology). **Source object type:** urn:oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject:Service **Target object type:** urn:x-ogc:specification:csw-ebrim:ObjectType:ServiceGrounding
*:DescribedBy	Associates a Service offer with a description of its computational characteristics and semantic content of requests (from the OWL-S 1.1 service ontology).

	Source object type: urn:oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject:Service **Target object type:** urn:x-ogc:specification:csw-ebrim:ObjectType:ServiceModel
*:Annotates	Associates an annotation resource with the registry object that it explains or evaluates. **Source object type:** urn:x-ogc:specification:csw-ebrim:ObjectType:Annotation **Target object type:** urn:oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject
*:RepositoryItemFor	Associates a source ExternalLink (that refers to an item in an external repository) with a target ExtrinsicObject. **Source object type:** urn:oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject:ExternalLink **Target object type:** urn:oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject:ExtrinsicObject
*:GraphicOverview	Associates a source Dataset with an Image that illustrates or summarizes the data (e.g., a browsing aid). **Source object type:** urn:x-ogc:specification:csw-ebrim:ObjectType:Dataset **Target object type:** urn:x-ogc:specification:csw-ebrim:ObjectType:Image

B.3 Classification schemes

The following classification schemes are included in the Basic package:

- Services taxonomy (source: ISO 19119, Subclause 8.3)
- Slots (source: DCMI metadata terms[3])
- Country codes (source: ISO 3166-1)
- Geographical regions (source: UN Statistics Division) <http://unstats.un.org/unsd/methods/m49/m49regin.htm>
- Feature codes (source: DIGEST v2.1, Part 4) <http://www.digest.org/html/DIGEST_2-1_Part4_AnnexA.pdf>

B.4 Stored queries

All of these stored queries have csw:GetRecordsResponse as the response document element, and thus may be invoked in a GetRecords context. A “view” parameter may be included with any query: “brief”, “summary” (default), or “full” are recognized values as defined in Subclause 7.4.

[3] See <http://dublincore.org/documents/dcmi-terms/>.

a) `findServices` — Returns a list of rim:Service elements. The optional "serviceType" parameter is an absolute URI value that identifies the service type of interest (a node in the the ISO 19119 services taxonomy).

b) `listExtensionPackages` — Returns a list of all deployed extension packages as a sequence of rim:RegistryPackage elements.

c) `showStoredQueries` — Returns a list of available stored query definitions as a sequence of rim:AdhocQuery elements. The optional "id" parameter is an absolute URI value that refers to a given stored query; multiple values may be included.

d) `getVersionHistory` — Returns the list of registry objects comprising the version history for the registry object specified by the required "id" parameter; if multiple values are provided, all but the first one is ignored.

B.5 Slots

Numerous slots are predefined as nodes in the 'Slots' classification scheme; refinements that specialize a given term are included as child nodes. Unless stated otherwise all slots can be applied to any registry object, or any subtype if a parent object type is identified. When a slot is added to a registry object, its name must match the name of the corresponding classification node.

The Basic package adds slots for many of the DCMI metadata terms: the core elements plus a few of the refinements; these may all be applied to any registry object. For simple values the slotType property should reference either an XML Schema datatype or a classification scheme if the values are selected from a controlled vocabulary; for complex values slotType must reference a suitable type definition. The slots are listed in Table B.3.

Table B.3 — Slots defined in the Basic package

Slot name	Definition
http://purl.org/dc/elements/1.1/contributor	An entity responsible for making contributions to the content of the resource.
http://purl.org/dc/elements/1.1/coverage	The extent or scope of the content of the resource.
http://purl.org/dc/elements/1.1/creator	An entity primarily responsible for making the content of the resource.
http://purl.org/dc/elements/1.1/date	A date associated with an event in the life cycle of the resource.
http://purl.org/dc/elements/1.1/description	An account of the content of the resource.
http://purl.org/dc/elements/1.1/format	The physical or digital manifestation of the resource.
http://purl.org/dc/elements/1.1/identifier	An unambiguous reference to the resource within a given context.
http://purl.org/dc/elements/1.1/language	A language of the intellectual content of the

	resource.
http://purl.org/dc/elements/1.1/publisher	An entity responsible for making the resource available.
http://purl.org/dc/elements/1.1/relation	A reference to a related resource.
http://purl.org/dc/elements/1.1/rights	Information about rights held in and over the resource.
http://purl.org/dc/elements/1.1/source	A reference to a resource from which the present resource is derived.
http://purl.org/dc/elements/1.1/subject	The topic of the content of the resource.
http://purl.org/dc/elements/1.1/title	A name given to the resource.
http://purl.org/dc/elements/1.1/type	The nature or genre of the content of the resource.
http://purl.org/dc/terms/abstract	A summary of the content of the resource (refinement of description).
http://purl.org/dc/terms/modified	Date on which the resource was changed (refinement of date).
http://purl.org/dc/terms/spatial	Spatial characteristics of the intellectual content of the resource (refinement of coverage).
http://purl.org/dc/terms/temporal	Temporal characteristics of the intellectual content of the resource (refinement of coverage).
http://purl.org/dc/terms/valid	Date (often a range) of validity of a resource (refinement of date).

Annex C
(informative)

W3C WSDL interface description

```
<?xml version="1.0" encoding="UTF-8"?>
<wsd:description
  targetNamespace="http://www.opengis.net/csw-ebrim/wsdl/1.0.0"
  xmlns:tns="http://www.opengis.net/csw-ebrim/wsdl/1.0.0"
  xmlns:wrs="http://www.opengis.net/cat/wrs"
  xmlns:kvp="http://www.opengis.net/cat/wrs/kvp"
  xmlns:csw="http://www.opengis.net/cat/csw"
  xmlns:ows="http://www.opengeospatial.net/ows"
  xmlns:wsd="http://www.w3.org/2005/08/wsdl"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <wsd:documentation>
  W3C WSDL 2.0 interface descriptions for the CSW-ebRIM catalogue
  application profile.
  </wsd:documentation>

  <wsd:types>
    <xs:import
      namespace="http://www.opengis.net/cat/wrs"
      schemaLocation="http://schemas.opengeospatial.net/csw-
ebrim/1.0.0/wrs-publication.xsd" />
    <xs:import
      namespace="http://www.opengis.net/cat/csw"
      schemaLocation="http://schemas.opengeospatial.net/csw/2.0.1/CSW-
publication.xsd" />
    <xs:import
      namespace="http://www.opengeospatial.net/ows"
schemaLocation="http://schemas.opengeospatial.net/ows/1.0.0/owsAll.xsd"
/>

    <xs:schema id="wrs-kvp"
      targetNamespace="http://www.opengis.net/cat/wrs/kvp"
      xmlns:kvp="http://www.opengis.net/cat/wrs/kvp">

      <xs:annotation>
        <xs:documentation xml:lang="en">
        This schema declares message elements for GET requests or
        POST requests encoded as content type "application/x-www-form-
        urlencoded" (i.e., KVP-style encoding).

        1. Parameter names and values are escaped. Space characters are
           replaced by '+', and then reserved characters are percent-
           encoded as described in RFC 3986, section 2.2.
        2. the parameter name is separated from the value by the EQUALS
           SIGN character and name/value pairs are separated from each
           other by the AMPERSAND character. If multiple values are
           allowed they are separated using the COMMA character.
        </xs:documentation>
      </xs:annotation>
```

```
      <xs:group name="common-elements">
        <xs:sequence>
          <xs:element name="service" type="xs:anyURI"
            fixed="urn:x-ogc:specification:csw-ebrim:Service:OGC-
CSW:ebRIM" />
          <xs:element name="request" type="xs:string" />
        </xs:sequence>
      </xs:group>
      <xs:element name="GetCapabilities"
        type="kvp:GetCapabilitiesType"/>
      <xs:complexType name="GetCapabilitiesType">
       <xs:sequence>
          <xs:group ref="kvp:common-elements"/>
          <xs:element name="acceptVersions"
            type="kvp:AcceptVersionsType" minOccurs="0" />
          <xs:element name="sections"
            type="kvp:CommaSeparatedWordsType" minOccurs="0" />
          <xs:element name="updateSequence" type="xs:string"
            minOccurs="0" />
          <xs:element name="acceptFormats"
            type="kvp:CommaSeparatedValuesType" minOccurs="0" />
        </xs:sequence>
      </xs:complexType>
      <xs:element name="GetRecordById" type="kvp:GetRecordByIdType"/>
      <xs:complexType name="GetRecordByIdType">
       <xs:sequence>
          <xs:group ref="kvp:common-elements"/>
          <xs:element name="id" type="xs:anyURI" />
          <xs:element name="elementSet" type="kvp:ElementSetType"
            minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
      <xs:simpleType name="AcceptVersionsType">
        <xs:annotation>
          <xs:documentation xml:lang="en">
          Examples: "2.0.35", "1.1.0,1.1.1"
          </xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
          <xs:pattern value="(\d+\.\d?\d\.\d?\d,?)+" />
        </xs:restriction>
      </xs:simpleType>
      <xs:simpleType name="CommaSeparatedValuesType">
        <xs:annotation>
          <xs:documentation xml:lang="en">
          Examples: "OperationsMetadata", "application/xml,text/html"
          </xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
          <xs:pattern value="([\w\+\-/]+,?)+" />
        </xs:restriction>
      </xs:simpleType>
      <xs:simpleType name="ElementSetType">
        <xs:restriction base="xs:string">
          <xs:enumeration value="brief" />
          <xs:enumeration value="summary" />
          <xs:enumeration value="full" />
```

```
      </xs:restriction>
    </xs:simpleType>
  </xs:schema>
</wsd:types>

<wsd:interface name="OGCWebService">

  <wsd:fault name="InvalidRequestFault"
    element="ows:ExceptionReport">
    <wsd:documentation>
    The body of the request message is invalid or not well formed.
    The document element must include a child ows:Exception element
    with code wrs:InvalidRequest.
    </wsd:documentation>
  </wsd:fault>

  <wsd:operation name="GetCapabilities"
    pattern="http://www.w3.org/2005/08/wsdl/in-out"
    style="http://www.w3.org/2005/08/wsdl/style/iri"
    safe="true">
    <wsd:documentation>
    Uses the GET method with the "application/x-www-form-urlencoded"
    serialization format. Message elements are inserted into the
    Request-URI.
    </wsd:documentation>
    <wsd:input element="kvp:GetCapabilities" />
    <wsd:output element="wrs:Capabilities" />
    <wsd:outfault ref="tns:InvalidRequestFault" />
  </wsd:operation>

  <wsd:operation name="GetCapabilities-xml"
    pattern="http://www.w3.org/2005/08/wsdl/in-out"
    safe="true">
    <wsd:documentation>
    Uses the POST method with the "application/xml" serialization
    format.
    </wsd:documentation>
    <wsd:input element="csw:GetCapabilities" />
    <wsd:output element="wrs:Capabilities" />
    <wsd:outfault ref="tns:InvalidRequestFault" />
  </wsd:operation>
</wsd:interface>

<wsd:interface name="Discovery">

  <wsd:fault name="InvalidRequestFault"
    element="ows:ExceptionReport">
    <wsd:documentation>
    The body of the request message is invalid or not well formed.
    The document element must include a child Exception element with
    code wrs:InvalidRequest.
    </wsd:documentation>
  </wsd:fault>

  <wsd:operation name="GetRecords"
    pattern="http://www.w3.org/2005/05/wsdl/in-out"
    safe="true">
    <wsd:input element="csw:GetRecords" />
```



```
    <wsd:output element="csw:GetRecordsResponse" />
    <wsd:outfault ref="tns:InvalidRequestFault" />
  </wsd:operation>

  <wsd:operation name="GetRecordById"
    pattern="http://www.w3.org/2005/08/wsdl/in-out"
    style="http://www.w3.org/2005/08/wsdl/style/iri"
    safe="true">
    <wsd:documentation>
    Uses the GET method with the "application/x-www-form-urlencoded"
    serialization format. Message elements are inserted into the
    Request-URI.
    </wsd:documentation>
    <wsd:input element="kvp:GetRecordById" />
    <wsd:output element="csw:GetRecordByIdResponse" />
    <wsd:outfault ref="tns:InvalidRequestFault" />
  </wsd:operation>

  <wsd:operation name="GetRecordById-xml"
    pattern="http://www.w3.org/2005/08/wsdl/in-out"
    safe="true">
    <wsd:input element="csw:GetRecordById" />
    <wsd:output element="csw:GetRecordByIdResponse" />
    <wsd:outfault ref="tns:InvalidRequestFault" />
  </wsd:operation>

  <wsd:operation name="DescribeRecord"
    pattern="http://www.w3.org/2005/08/wsdl/in-out"
    safe="true">
    <wsd:input element="csw:DescribeRecord" />
    <wsd:output element="csw:DescribeRecordResponse" />
    <wsd:outfault ref="tns:InvalidRequestFault" />
  </wsd:operation>

  <wsd:operation name="GetDomain"
    pattern="http://www.w3.org/2005/08/wsdl/in-out"
    safe="true">
    <wsd:input element="csw:GetDomain" />
    <wsd:output element="csw:GetDomainResponse" />
    <wsd:outfault ref="tns:InvalidRequestFault" />
  </wsd:operation>

  <wsd:operation name="GetRepositoryItem"
    pattern="http://www.w3.org/2005/08/wsdl/in-out"
    style="http://www.w3.org/2005/08/wsdl/style/iri"
    safe="true">
    <wsd:documentation>
    Uses the GET method with the "application/x-www-form-urlencoded"
    serialization format. Message elements are inserted into the
    Request-URI.
    </wsd:documentation>
    <wsd:input element="kvp:GetRepositoryItem" />
    <wsd:output element="#other">
      <wsd:documentation>
      The entity-body must be an instance of a registered MIME media
      type. See http://www.iana.org/assignments/media-types/.
      </wsd:documentation>
    </wsd:output>
```

```
      <wsd:outfault ref="tns:InvalidRequestFault" />
    </wsd:operation>
  </wsd:interface>

  <wsd:interface name="Publication">

    <wsd:fault name="InvalidRequestFault"
      element="ows:ExceptionReport">
      <wsd:documentation>
      The body of the request message is invalid or not well formed.
      The document element must include a child ows:Exception element
      with code wrs:InvalidRequest.
      </wsd:documentation>
    </wsd:fault>

    <wsd:fault name="TransactionFailedFault"
      element="ows:ExceptionReport">
      <wsd:documentation>
      The requested transaction could not be completed for some reason.
      The document element must include a child ows:Exception element
      with code wrs:TransactionFailed.
      </wsd:documentation>
    </wsd:fault>

    <wsd:operation name="Transaction"
      pattern="http://www.w3.org/2005/08/wsdl/in-out">

      <wsd:input element="csw:Transaction"/>
      <wsd:output element="csw:TransactionResponse"/>
      <wsd:outfault ref="tns:InvalidRequestFault" />
      <wsd:outfault ref="tns:TransactionFailedFault" />
    </wsd:operation>

    <wsd:operation name="Harvest"
      pattern="http://www.w3.org/2005/08/wsdl/in-out">

      <wsd:input element="csw:Harvest"/>
      <wsd:output element="csw:HarvestResponse"/>
      <wsd:outfault ref="tns:InvalidRequestFault" />
      <wsd:outfault ref="tns:TransactionFailedFault" />
    </wsd:operation>
  </wsd:interface>

</wsd:description>
```


Annex D
(informative)

Examples

D.1 Service capabilities document

```
<?xml version="1.0" encoding="UTF-8"?>
<wrs:Capabilities
  xmlns:wrs="http://www.opengis.net/cat/wrs"
  xmlns:csw="http://www.opengis.net/cat/csw"
  xmlns:ows="http://www.opengeospatial.net/ows"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  version="0.10.2">

  <ows:ServiceIdentification>
    <ows:Title>ACME Catalogue Service</ows:Title>
    <ows:Abstract>
    A web-based catalogue service that implements the CSW-ebRIM profile
    of the OGC Catalogue 2.0 specification.
    </ows:Abstract>
    <ows:Keywords>
      <ows:Keyword>registry</ows:Keyword>
      <ows:Keyword>catalogue</ows:Keyword>
      <ows:Keyword>ebRIM</ows:Keyword>
    </ows:Keywords>
    <ows:ServiceType>
      urn:x-ogc:service:catalogue:csw-ebrim
    </ows:ServiceType>
    <ows:ServiceTypeVersion>0.10.2</ows:ServiceTypeVersion>
    <ows:Fees>NONE</ows:Fees>
    <ows:AccessConstraints>
    Basic authentication (RFC 2617) is required for all transaction
requests.
    </ows:AccessConstraints>
  </ows:ServiceIdentification>

  <ows:ServiceProvider>
    <ows:ProviderName>Acme Systems, Inc.</ows:ProviderName>
    <ows:ProviderSite xlink:type="simple"
      xlink:title="Corporate web site"
      xlink:href="http://www.acme.org"/>
    <ows:ServiceContact>
      <ows:IndividualName>Phineas Fogg</ows:IndividualName>
      <ows:PositionName>System administrator</ows:PositionName>
      <ows:ContactInfo>
        <ows:Phone>
          <ows:Voice>+1 444 555-4444</ows:Voice>
          <ows:Facsimile>+1 444 555-4445</ows:Facsimile>
        </ows:Phone>
        <ows:Address>
          <ows:DeliveryPoint>ACME Systems</ows:DeliveryPoint>
```



```
          <ows:City>Erewhon</ows:City>
          <ows:AdministrativeArea>NT</ows:AdministrativeArea>
          <ows:PostalCode>ABC123</ows:PostalCode>
          <ows:Country>Ruritania</ows:Country>
          <ows:ElectronicMailAddress>
          pfogg@acme.com
          </ows:ElectronicMailAddress>
        </ows:Address>
        <ows:HoursOfService>04:00-06:00 CEST</ows:HoursOfService>
        <ows:ContactInstructions>Please use email for all
inquiries.</ows:ContactInstructions>
      </ows:ContactInfo>
      <ows:Role>pointOfContact</ows:Role>
    </ows:ServiceContact>
  </ows:ServiceProvider>

  <ows:OperationsMetadata>
    <ows:Operation name="GetCapabilities">
      <ows:DCP>
        <ows:HTTP>
          <ows:Get
xlink:href="http://catalogue.demo.acme.com/catalogue/csw"/>
        </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="sections">
        <ows:Value>ServiceIdentification</ows:Value>
        <ows:Value>ServiceProvider</ows:Value>
        <ows:Value>OperationsMetadata</ows:Value>
        <ows:Value>Filter_Capabilities</ows:Value>
        <ows:Value>ServiceFeatures</ows:Value>
        <ows:Value>ServiceProperties</ows:Value>
      </ows:Parameter>
    </ows:Operation>
    <ows:Operation name="GetRecords">
      <ows:DCP>
        <ows:HTTP>
          <ows:Post
xlink:href="http://catalogue.demo.acme.com/catalogue/csw"/>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
    <ows:Operation name="GetRecordById">
      <ows:DCP>
        <ows:HTTP>
          <ows:Get
xlink:href="http://catalogue.demo.acme.com/catalogue/csw"/>
        </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="Id">
        <ows:Value/>
      </ows:Parameter>
    </ows:Operation>
    <ows:Operation name="DescribeRecord">
      <ows:DCP>
        <ows:HTTP>
          <ows:Post
xlink:href="http://catalogue.demo.acme.com/catalogue/csw"/>
        </ows:HTTP>
```



```
      </ows:DCP>
    </ows:Operation>
    <ows:Operation name="GetDomain">
      <ows:DCP>
        <ows:HTTP>
          <ows:Post
xlink:href="http://catalogue.demo.acme.com/catalogue/csw"/>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
    <ows:Operation name="GetRepositoryItem">
      <ows:DCP>
        <ows:HTTP>
          <ows:Get
xlink:href="http://catalogue.demo.acme.com/catalogue/csw"/>
        </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="id">
        <ows:Value/>
      </ows:Parameter>
    </ows:Operation>
    <ows:Operation name="Harvest">
      <ows:DCP>
        <ows:HTTP>
          <ows:Post
xlink:href="http://catalogue.demo.acme.com/catalogue/csw"/>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
    <ows:Operation name="Transaction">
      <ows:DCP>
        <ows:HTTP>
          <ows:Post
xlink:href="http://catalogue.demo.acme.com/catalogue/csw"/>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
    <ows:Parameter name="service">
      <ows:Value>urn:x-ogc:service:catalogue:csw-ebrim</ows:Value>
    </ows:Parameter>
  </ows:OperationsMetadata>

  <ogc:Filter_Capabilities
    xmlns:gml="http://www.opengis.net/gml">
    <ogc:Spatial_Capabilities>
      <ogc:GeometryOperands>
        <ogc:GeometryOperand>gml:Envelope</ogc:GeometryOperand>
        <ogc:GeometryOperand>gml:Point</ogc:GeometryOperand>
        <ogc:GeometryOperand>gml:LineString</ogc:GeometryOperand>
        <ogc:GeometryOperand>gml:Polygon</ogc:GeometryOperand>
      </ogc:GeometryOperands>
      <ogc:SpatialOperators>
        <ogc:SpatialOperator name="BBOX"/>
        <ogc:SpatialOperator name="Equals"/>
        <ogc:SpatialOperator name="Disjoint"/>
        <ogc:SpatialOperator name="Intersect"/>
        <ogc:SpatialOperator name="Touches"/>
        <ogc:SpatialOperator name="Crosses"/>
```

```
        <ogc:SpatialOperator name="Within"/>
        <ogc:SpatialOperator name="Contains"/>
        <ogc:SpatialOperator name="Overlaps"/>
        <ogc:SpatialOperator name="Beyond"/>
      </ogc:SpatialOperators>
    </ogc:Spatial_Capabilities>
    <ogc:Scalar_Capabilities>
      <ogc:LogicalOperators/>
      <ogc:ComparisonOperators>
        <ogc:ComparisonOperator>LessThan</ogc:ComparisonOperator>
        <ogc:ComparisonOperator>GreaterThan</ogc:ComparisonOperator>

<ogc:ComparisonOperator>LessThanEqualTo</ogc:ComparisonOperator>

<ogc:ComparisonOperator>GreaterThanEqualTo</ogc:ComparisonOperator>
        <ogc:ComparisonOperator>EqualTo</ogc:ComparisonOperator>
        <ogc:ComparisonOperator>NotEqualTo</ogc:ComparisonOperator>
        <ogc:ComparisonOperator>Like</ogc:ComparisonOperator>
        <ogc:ComparisonOperator>Between</ogc:ComparisonOperator>
        <ogc:ComparisonOperator>NullCheck</ogc:ComparisonOperator>
      </ogc:ComparisonOperators>
      <ogc:ArithmeticOperators>
        <ogc:SimpleArithmetic />
      </ogc:ArithmeticOperators>
    </ogc:Scalar_Capabilities>
  </ogc:Filter_Capabilities>

  <wrs:ServiceFeatures>
    <wrs:feature
      name="http://www.opengis.net/cat/wrs/features/deep-search">
      <wrs:property
        name="http://www.opengis.net/cat/wrs/properties/mime-types">
        <wrs:value>application/xml</wrs:value>
        <wrs:value>text/xml</wrs:value>
      </wrs:property>
    </wrs:feature>
    <wrs:feature
      name="http://www.opengis.net/cat/wrs/features/audit-trail" />
    <wrs:feature
      name="http://www.opengis.net/cat/wrs/features/version-control">
      <wrs:property
        name="http://www.opengis.net/cat/wrs/properties/object-types">
        <wrs:value>urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:Service</wrs:value>
        <wrs:value>urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject</wrs:value>
      </wrs:property>
    </wrs:feature>
  </wrs:ServiceFeatures>

  <wrs:ServiceProperties>
    <wrs:property
      name="http://www.opengis.net/cat/wrs/properties/extension-
packages">
      <wrs:value>urn:x-ogc:specification:csw-ebrim:ext-
pkg:Basic</wrs:value>
    </wrs:property>
    <wrs:property
```



```
      name="http://www.opengis.net/cat/wrs/properties/harvest-
protocols">
      <wrs:value>http</wrs:value>
    </wrs:property>
    <wrs:property
      name="http://www.opengis.net/cat/wrs/properties/query-languages">
      <wrs:value>http://www.opengis.net/ogc</wrs:value>
      <wrs:value>http://www.w3.org/TR/xpath</wrs:value>
    </wrs:property>
    <wrs:property
      name="http://www.opengis.net/cat/wrs/properties/mime-types">
      <wrs:value>application/xml</wrs:value>
      <wrs:value>text/xml</wrs:value>
    </wrs:property>
    <!-- duplicates info in ogc:GeometryOperands? -->
    <wrs:property
      name="http://www.opengis.net/cat/wrs/properties/geometry-types">
      <wrs:value>gml:Point</wrs:value>
    </wrs:property>
    <wrs:property
      name="http://www.opengis.net/cat/wrs/properties/temporal-ref-
systems">
      <wrs:value>urn:x-ogc:def:trs:ISO-8601</wrs:value>
    </wrs:property>
    <wrs:property
      name="http://www.opengis.net/cat/wrs/properties/spatial-ref-
systems">
      <wrs:value>urn:x-ogc:def:crs:EPSG:4326</wrs:value>
    </wrs:property>
  </wrs:ServiceProperties>

  <wrs:WSDL-services xlink:type="simple"
    xlink:href="http://catalogue.demo.acme.com/catalogue/wsdl"
    xlink:title="Available service endpoints (WSDL 2.0)"
    xlink:role="http://www.w3.org/2005/08/wsdl" />

</wrs:Capabilities>
```

D.2 Spatial query (BBOX operator)

```
<?xml version="1.0" encoding="UTF-8"?>
<GetRecords
  xmlns="http://www.opengis.net/cat/csw"
  startPosition="1"
  maxRecords="20"
  resultType="results">

  <Query typeNames="rim:Service rim:Slot=s"
    xmlns:rim="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0"
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:ogc="http://www.opengis.net/ogc">
    <ElementSetName typeNames="rim:Service">summary</ElementSetName>
    <Constraint version="1.1">
      <ogc:Filter>
        <ogc:And>
```



```
          <ogc:PropertyIsEqualTo>
            <ogc:PropertyName>rim:Service/$s/@name</ogc:PropertyName>
            <ogc:Literal>http://purl.org/dc/terms/spatial</ogc:Literal>
          </ogc:PropertyIsEqualTo>
          <ogc:BBOX>
            <ogc:PropertyName>
            rim:Service/$s/wrs:ValueList/wrs:AnyValue
            </ogc:PropertyName>
            <gml:Envelope>
              <gml:lowerCorner>48.86 -124.18</gml:lowerCorner>
              <gml:upperCorner>49.98 -122.32</gml:upperCorner>
            </gml:Envelope>
          </ogc:BBOX>
        </ogc:And>
      </ogc:Filter>
    </Constraint>
  </Query>
</GetRecords>
```


Bibliography

[1] *URNs of definitions in ogc namespace*, OGC document 05-010, available [online]: <https://portal.opengeospatial.org/files/?artifact_id=8814>.

[2] Dublin Core Metadata Initiative, *DCMI Metadata Terms*, available [online]: <http://dublincore.org/documents/dcmi-terms/>.

[3] IETF RFC 3406, *Uniform Resource Names (URN) Namespace Definition Mechanisms*, Best Current Practice (October 2002), available [online]: <http://www.apps.ietf.org/rfc/rfc3406.html>.