Open Geospatial Consortium Inc.

Date: 2008-02-29

Reference number of this document: OGC 07-110r2

Version: 1.0.0

Category: OpenGIS® Extension

Editor(s): R. Martell

CSW-ebRIM Registry Service - Part 1: ebRIM profile of CSW

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® Extension
Document subtype:	Class 2 profile
Document stage:	Approved Standard
Document language:	English

Contents

Page

Figures

Page

Tables

Page

i. Preface

The OGC Catalogue Services 2.0.2 specification (OGC 07-006r1) establishes a general framework for implementing catalogue services that can be used to meet the needs of stakeholders in a wide variety of application domains. This profile is based on the HTTP protocol binding described in Clause 10 of the Catalogue 2.0.2 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.

Suggested additions, changes, and comments on this draft report are welcome and encouraged. Such suggestions may be submitted by email message or by making suggested changes in an edited copy of this document.

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 in order 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
R. Martell <rmartell AT galdosinc DOT com>	Galdos Systems, Inc.
F. Najmi < farrukh AT wellfleetsoftware DOT com>	Wellfleet Software Corporation
O. Newell <olivern AT ll DOT mit DOT edu>	MIT Lincoln Laboratory
R. Primavera <renato DOT primavera AT ionicsoft DOT com>	Leica Geosystems Geospatial Imaging, LLC
M.L. Vautier <marie-lise DOT vautier AT ign DOT fr>	Institut Geographique National (IGN)
P. Vretanos <pvretano AT cubewerx DOT com>	CubeWerx

iv. Revision history

Date	Release	Editor	Primary clauses modified	Description
2007-11-21	1.0.0-rc1	R. Martell		Initial release of candidate standard.
2007-12-12	1.0.0-rc2	R. Martell		RWG-approved standard.
2008-02-29	1.0.0	R. Martell		Final approved standard.

v. Changes to the OGC Abstract Specification

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

Foreword

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

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

This document cancels and replaces OGC document 05-025r3 and all previous revisions of OGC 07-110 in their entirety. It depends primarily on the following specifications:

- OGC Catalogue Services Specification 2.0.2 [OGC 07-006r1]
- OASIS ebXML Registry Information Model v3.0
- OWS Common Implementation Specification 1.0 [OGC 05-008]
- Filter Encoding Implementation Specification 1.1 [OGC 04-095]
- OGC Geography Markup Language (GML) 3.1 [OGC 03-105r1]
- IETF RFC 2616 (Hypertext Transfer Protocol -- HTTP/1.1)

The CSW-ebRIM specification consists of the following parts:

- OGC 07-110r2, *CSW-ebRIM Registry Service, Part 1: ebRIM profile of CSW*
- OGC 07-144r2, *CSW-ebRIM Registry Service, Part 2: Basic extension package*

Introduction

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 registry 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).

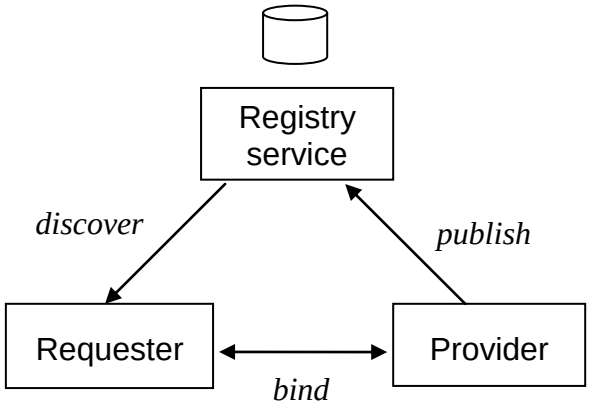

Figure 1 – Essential interactions in a service-oriented architecture

The CSW-ebRIM Registry Service profile is based on the HTTP protocol binding (the CSW part) documented in Clause 10 of the OGC Catalogue Services Specification (version 2.0.2, OGC 07-006r1). The profile imposes some constraints on the the use of the base specifications and introduces additional search, retrieval, and registry management capabilities. It provides facilities for advertising and discovering and 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 treated as metadata in another.

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.

CSW-ebRIM Registry Service - Part 1: ebRIM profile of CSW

1 Scope

This OGC® document specifies the CSW-ebRIM Registry Service: a profile of the CSW part (Clause 10) of the *OpenGIS® Catalogue Service Implementation Specification* (v2.0.2, OGC-07-006r1). It joins the CSW interfaces to the OASIS ebXML registry information model (ebRIM 3.0) so as to provide a general and flexible web-based registry service that enables users—human or software agents—to locate, access, and make use of resources in an open, distributed system; it provides facilities for retrieving, storing, and managing many kinds of resource descriptions. An extension mechanism permits registry content to be tailored for more specialized application domains.

2 Compliance

2.1 Conformance requirements

This specification defines two levels of conformance. Any product claiming conformance with this specification at one of these levels shall pass all applicable tests specified in the abstract test suite (ATS) appearing in Annex A. In addition to satisfying the requirements stipulated for a given conformance level, a conforming implementation must also satisfy the relevant requirements in all normative base specifications.

2.2 Conformance level 1

Level 1 covers search and retrieval capabilities; compliance at this level is mandatory. The following general service capabilities shall be assessed:

a) GetCapabilities (see Clause 8)

b) DescribeRecord (see Clause 9)

c) GetRecords (see Clause 10)

d) GetRecordById (see Clause 11)

e) GetRepositoryItem (see Clause 12)

f) GetDomain (see Clause 13)

g) Predefined queries (see Clause 16)

2.3 Conformance level 2

Level 2 covers publication facilities used to manage registry content; it includes all requirements for Level 1. The following additional service capabilities—at least one of which must be implemented—shall be assessed at this level:

a) Harvest (see Clause 14)

b) Transaction (see Clause 15)

3 Normative references

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IETF RFC 2387, *The MIME Multipart/Related Content-type*. Available from: <http://tools.ietf.org/html/rfc2387>.

IETF RFC 2392, *Content-ID and Message-ID Uniform Resource Locators*. Available from: <http://tools.ietf.org/html/rfc2392>.

IETF RFC 2616, *Hypertext Transfer Protocol -- HTTP/1.1*. Available from: <http://tools.ietf.org/html/rfc2616>.

IETF RFC 3902, *The "application/soap+xml" media type*. Available from: <http://tools.ietf.org/html/rfc3902>

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

OASIS regrep-rim-3.0-os, *ebXML Registry Information Model, Version 3.0.* Available from: <http://docs.oasis-open.org/regrep/v3.0/specs/regrep-rim-3.0-os.pdf>.

OGC 07-006r1, *OpenGIS® Catalogue Services Specification, Version 2.0.2.* Available from: <http://portal.opengeospatial.org/files/?artifact_id=20555>.

OGC 04-095, *Filter Encoding Implementation Specification,Version 1.1.0.* Available from: <http://portal.opengeospatial.org/files/?artifact_id=8340>.

OGC 05-008, *OpenGIS® Web Services Common Specification, Version 1.0.0.* Available from: <http://portal.opengeospatial.org/files/?artifact_id=8798>.

OGC 03-105r1, *OpenGIS® Geography Markup Language (GML) Implementation Specification, Version 3.1.1* (ISO/CD 19136). Available from: <http://portal.opengeospatial.org/files/?artifact_id=4700>.

W3C SOAP-1, *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. Available from: <http://www.w3.org/TR/soap12-part1/>.

W3C SOAP-2, *SOAP Version 1.2 Part 2: Adjuncts (Second Edition)*. Available from: <http://www.w3.org/TR/soap12-part2/>.

W3C XOP, *XML-binary Optimized Packaging.* Available from: <http://www.w3.org/TR/xop10/>.

W3C XPATH, *XML Path Language (XPath) Version 1.0.* Available from: <http://www.w3.org/TR/xpath>.

In addition to this document, the specification includes the normative XML Schema document contained in Annex B.

4 Terms and definitions

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

4.1
extension package
registry package containing a 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.

EXAMPLE A Geodesy package for CRS definitions; a Gazetteer package for feature registries

4.2
extrinsic object
registry object that describes a repository item managed by the registry.

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
repository item
resource formatted in accord with an Internet media type.

EXAMPLE XML Schema, specification document, graphic image, audio recording, video clip.

4.5
resource
network data object or service that can be identified by a URI. [RFC 2616]

4.6
version
resource that contains a copy of a particular state of a version-controlled registry object or repository item.

4.7
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 Clause 5.1 of the OWS Common Implementation Specification [OGC 05-008] apply to this document, plus the following abbreviated terms.

ATC	Abstract Test Case
ATS	Abstract Test Suite
CRS	Coordinate Reference System
ebRIM	ebXML Registry Information Model
IRI	Internationalized Resource Identifier
IUT	Implementation Under Test
TLS	Transport Layer Security
URI	Uniform Resource Identifier
URN	Uniform Resource Name
WSDL	Web Services Description Language
XOP	XML-binary Optimized Packaging

5.2 Namespaces

Several prefixes are used throughout this document to designate XML namespaces. Table 1 lists the namespaces used in this document and the specifications in which they are defined; these bindings shall be assumed in the absence of an explicit declaration. The prefixes are **not** normative and are merely employed 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/1.0	CSW-ebRIM profile (OGC 07-110r2)

Prefix	Namespace URI	Specification
rim	urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0	OASIS ebRIM 3.0
csw	http://www.opengis.net/cat/csw/2.0.2	OGC Catalogue Services 2.0.2 (OGC 07-006r1)
ows	http://www.opengeospatial.net/ows	OWS Common 1.0 (OGC 05-008)
ogc	http://www.opengis.net/ogc	Filter 1.1 (OGC 04-095)
gml	http://www.opengis.net/gml	GML 3.1 (OGC 03-105r1)
dc	http://purl.org/dc/elements/1.1/	Namespace Policy for the DCMI[a]
wsdl	http://www.w3.org/ns/wsdl	W3C WSDL 2.0 Part 1
wsdi	http://www.w3.org/ns/wsdl-instance	W3C WSDL 2.0 Part 1
xlink	http://www.w3.org/1999/xlink	W3C XLink 1.0
xml	http://www.w3.org/XML/1998/namespace	Namespaces in XML 1.0
xop	http://www.w3.org/2004/08/xop/include	W3C XOP

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

6 Service overview

6.1 Essential capabilities

The essential purpose of a registry service is to enable a human or software agent to locate, access, and use resources of interest. The metadata repository managed by the catalogue can be used to store resource descriptions that conform to any standard Internet media type, including: service descriptions, data descriptions, schemas, thumbnail representations of remotely sensed images, specification documents, or a summary map of nodes in a sensor network. Furthermore, 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 provided by the service.

The CSW-ebRIM Registry Service is a specialized catalogue service based on the CSW part (Clause 10) of the OGC Catalogue Services 2.0.2 specification. In essence it joins the CSW interfaces to the OASIS ebXML registry information model (ebRIM 3.0). The principal specification dependencies are illustrated in Figure 2.

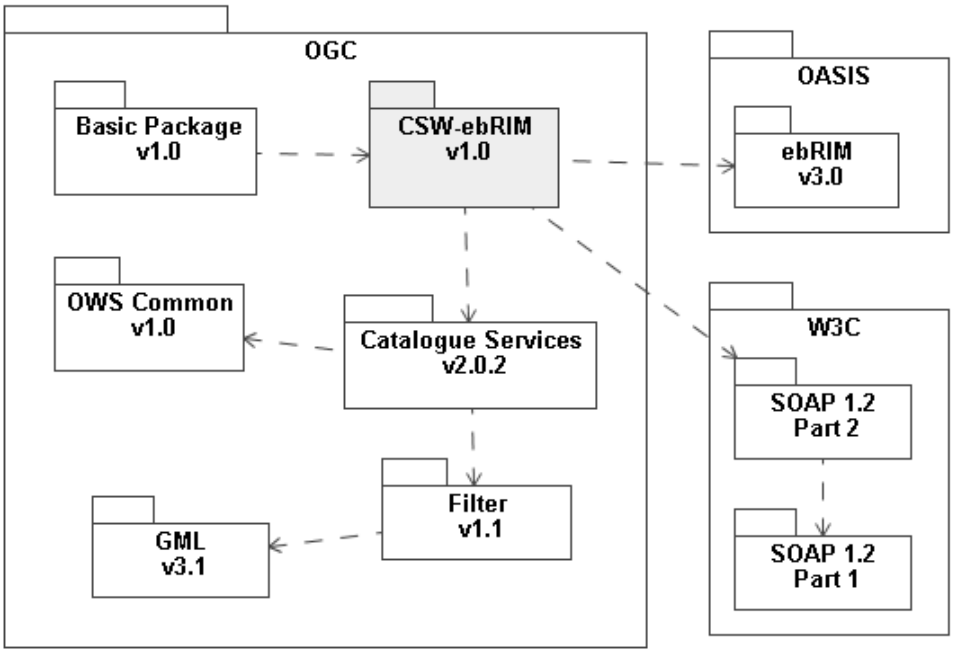

Figure 2 – Specification dependencies

A CSW-ebRIM implementation supports the requests summarized below, most of which are common to all CSW-based catalogue services:

a) GetCapabilities – This request allows a client to retrieve a resource that describes the essential capabilities and non-computational characteristics of the service.

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

c) GetRecords – This request allows a client to search the registry and retrieve all or some of the matching items.

d) GetRecordById – This request allows a client to retrieve a representation of one or more registry objects by identifier.

e) GetDomain – This request produces a description of the value domain for a specified data element or request parameter.

f) GetRepositoryItem – This requests allows a client to retrieve the repository item corresponding to some extrinsic object.

g) Harvest – This request enables a 'pull' style of publication whereby a resource is retrieved from some remote location and inserted into the catalogue.

h) Transaction – This request allows a client to directly insert, update, or delete catalogue content.

Figure 3 is a simple UML component diagram summarizing the CSW-ebRIM interfaces.

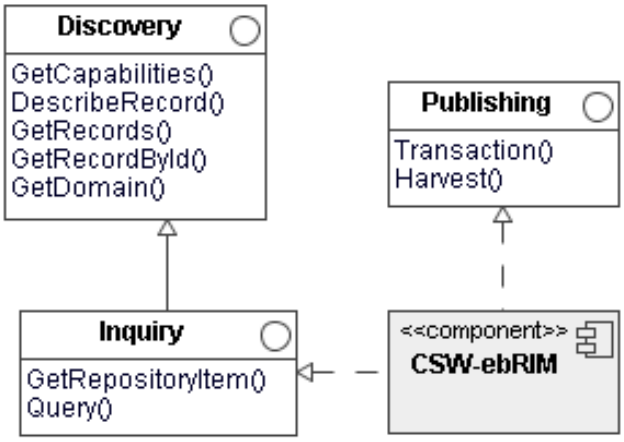

Figure 3 – CSW-ebRIM interfaces

6.2 Information model: ebRIM 3.0

The information model is based on version 3.0 of the OASIS ebXML Registry Information Model (ebRIM 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 4 is intended to provide a condensed overview of the information model—it is not normative. The OASIS standard defines a normative XML Schema for representing registry objects.

A rim:RegistryObject element—or any element that can substitute for it—may be included in a CSW message in any context where a representation of a csw:Record is allowed. The ebRIM representations shall be included if the value of the outputSchema attribute specified in a request matches the target namespace of the ebRIM schema ("urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0"). Furthermore, a full view of an ebRIM registry object representation shall be used in all insert and update statements appearing within a csw:Transaction request.

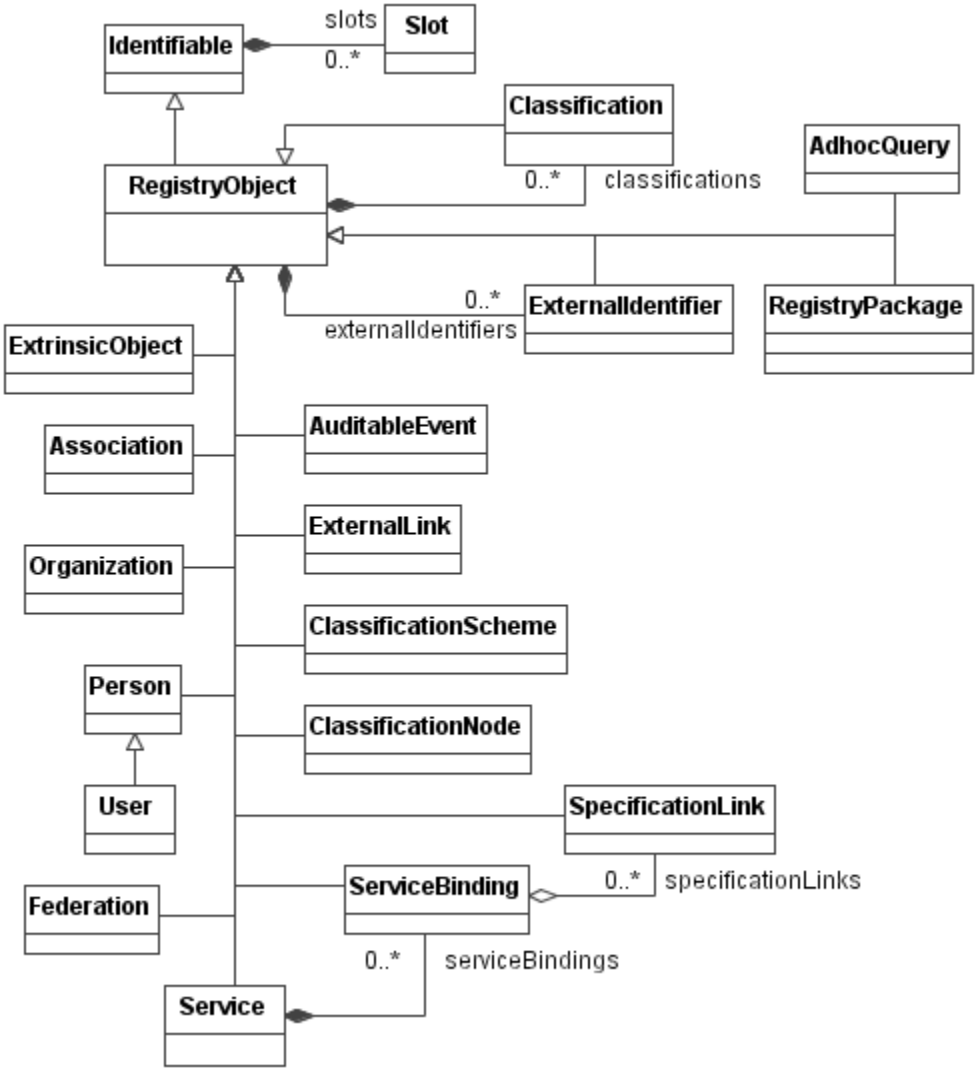

Figure 4 -- High-level view of ebRIM 3.0

The ebRIM 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 may be defined within an extension package. An extension package is a RegistryPackage containing canonical resources that constrain the content of the registry (see clause 16).

7 General aspects

7.1 Use of HTTP methods

The HTTP/1.1 specification [RFC 2616] defines eight methods for manipulating and retrieving representations of resources. Only the GET and POST methods are supported in this application profile. Service requests are bound to HTTP methods as indicated in Table 2.

Table 2 — HTTP method bindings

Operation name	HTTP method binding(s)
GetCapabilities	GET, POST (optional)
GetRecords	POST, GET (optional)
GetRecordById	GET, POST (optional)
DescribeRecord	GET, POST (optional)
GetDomain	GET, POST (optional)
GetRepositoryItem	GET
Transaction	POST
Harvest	POST

7.2 Use of HTTP message headers

The HTTP/1.1 specification defines a variety of message header fields that are 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; all request and response messages shall correctly indicate the content type of the message (if not empty). The following message content types shall be recognized by all conforming implementations:

a) "application/xml" – the body contains an XML request or response entity.

b) "application/x-www-form-urlencoded" – the body contains a KVP-style encoding of request parameters.

c) "application/soap+xml" – the body contains a SOAP 1.2 envelope (see cl. 7.5).

d) "multipart/related" – the message is a compound object containing related parts structured in accord with RFC 2387.

Use of the other header fields appearing in Table 3 is strongly recommended. If any of these headers are included in a request message, they shall not be ignored.

Table 3 — HTTP message header fields

Header name	Description
Accept	Request header that specifies a range of acceptable MIME media types for the response[a].

Header name	Description
Accept-Charset	Request header that indicates what character sets are acceptable for the response[b]
Accept-Encoding	Request header that indicates acceptable content codings for the response (e.g. gzip)
Accept-Language	Request header that expresses a preferred natural language for the response.
Content-Encoding	Entity header that identifies an encoding (e.g. gzip) that has modified the media type of the entity-body in a request or response message.
Content-Type	Entity header that indicates the MIME media type of the enclosed entity-body in a request or response.
Last-Modified	Entity header that indicates the date and time at which the enclosed response entity 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>.

If a service request includes the *outputFormat* parameter, this shall override any preferences expressed in the Accept request header field.

7.3 Repository items

A CSW-ebRIM service can be used to manage arbitrary media resources such as schemas, photographs, documents, audio recordings, video clips, and so forth. These resources are collectively referred to as repository items. The ebRIM model defines an ExtrinsicObject type to describe an internally managed repository item that is formatted according to some Internet media type, as indicated by the value of the mimeType attribute; this value shall correspond to a registered MIME media type.

NOTE The IANA MIME type registry is available online at <http://www.iana.org/assignments/media-types/>.

A repository item shall be described using a wrs:ExtrinsicObject element. The presence of the wrs:repositoryItemRef or wrs:repositoryItem element indicates the existence of a repository item; if no repository item exists, neither element shall be included.

Note The rim:ContentVersionInfo element is created by the service if versioning of repository items is supported. However, such versioning behaviour is beyond the scope of this specification.

The service capabilities document shall list all acceptable media types as values of the global "media-types" constraint in the ows:OperationsMetadata section. If a service can store the repository item being submitted, it creates a corresponding wrs:ExtrinsicObject item; otherwise an exception is raised.

7.4 Exception codes

The *OWS Common* specification defines a number of general exception codes that shall be supported by all implementations (OGC 05-008, cl. 8.3). This profile defines additional exception codes and also requires the proper use of HTTP status codes, as shown in Table 3.

Table 4 — CSW-ebRIM exception codes

Code	Reason	Status code
wrs:InvalidRequest	The request message is invalid for some reason.	400
wrs:NotImplemented	The request is not supported by the service.	500
wrs:NotFound	The requested resource does not exist or could not be found.	404
wrs:NotSupported	A service option or media type is unsupported.	415
wrs:TransactionFailed	The requested transaction could not be completed.	500

If the Request-URI contains any unrecognized query parameters, these may be ignored and an exception shall not be raised.

7.5 Registry object views

The OGC Catalogue Services specification (OGC 07-006r1) distinguishes three abstract property sets—or views—that provide differing levels of detail about a catalogue item: brief, summary, and full. These abstract views are mapped to the ebRIM schema as indicated in Table 5.

Table 5 — Registry object views

View name	ebRIM information items
brief	rim:RegistryObject/@id rim:RegistryObject/@lid[a] 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) [b] rim:RegistryObject/rim:Description (in preferred languages) [b]
full	Complete representation.
The brief and summary views map to reduced rim:RegistryObject representations for any object type. A full view yields the element information item corresponding to the actual object type. a The value of the @lid attribute implicitly identifies the "version history" resource for a registry	

object. Its value is set by the service to coincide with the @id value of the original registry object.
b As specified by the value of the the Accept-Language request header field (if present).

Example 1 Brief view of ebRIM registry object.

```
<rim:RegistryObject
 id="urn:uuid:722fdf66-9222-11dc-8314-0800200c9a66"
 lid="urn:uuid:722fdf66-9222-11dc-8314-0800200c9a66"
 status="urn:oasis:names:tc:ebxml-regrep:StatusType:Submitted"
 objectType="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:Service">
  <rim:VersionInfo versionName=" " />
</rim:RegistryObject>
```

Example 2 Summary view of ebRIM registry object.

```
<rim:RegistryObject
 id="urn:uuid:722fdf66-9222-11dc-8314-0800200c9a66"
 lid="urn:uuid:722fdf66-9222-11dc-8314-0800200c9a66"
 status="urn:oasis:names:tc:ebxml-regrep:StatusType:Submitted"
 objectType="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:Service">
  <rim:Slot name="Modified">
    <rim:ValueList>
      <rim:Value>2007-11-14T12:18:26-08:00</rim:Value>
    </rim:ValueList>
  </rim:Slot>
  <rim:Name>
    <rim:LocalizedString xml:lang="en" value="Name">
  </rim:Name>
  <rim:Description>
    <rim:LocalizedString xml:lang="en" value="Description">
  </rim:Description>
  <rim:VersionInfo versionName=" " />
</rim:RegistryObject>
```

7.6 Mapping ebRIM objects to CSW record representations

Clause 10 of the OGC Catalogue Services specification defines a simple retrieval record format that is common to all CSW-based catalogue services. The CSW record representation shall be included in response if the requested outputSchema is “http://www.opengis.net/cat/csw/2.0.2”.

Table 8 shows how ebRIM registry objects are mapped to the CSW record representation. Some ebRIM information items may have multiple occurrences; as a consequence, the CSW properties to which they correspond may occur more than once in the resulting record.

Table 6 — Mapping ebRIM information items to csw:Record properties

ebRIM information item(s)	CSW record property[a]
rim:RegistryObject/@id	dc:identifier[b]
rim:RegistryObject/rim:ExternalIdentifier/@value	dc:identifier
rim:RegistryObject/@objectType	dc:type
rim:RegistryObject/rim:Name/rim:LocalizedString/@value	dc:title {1..*}
rim:RegistryObject/rim:Description/rim:LocalizedString/@value	dc:description {1..*}
rim:ExtrinsicObject/@mimeType	dc:format
rim:RegistryObject/rim:Slot[@slotType = "*:GM_Envelope"]/rim:ValueList/rim:Value[c]	ows:BoundingBox {1..*}
rim:RegistryObject/rim:Slot[@name = "http://purl.org/dc/elements/1.1/subject"]/rim:ValueList/rim:Value	dc:subject {1..*}
rim:ExternalIdentifier/@registryObject	dc:relation

a The cardinality constraint {1..*} indicates that the element may appear more than once.

b The first identifier (in document order) is taken as the principal identifier; remaining identifiers correspond to external identifiers.

C The abbreviated slotType value expands to "urn:ogc:def:dataType:ISO-19107:GM_Envelope".

The csw:Record/csw:AnyText element is intended solely to support full-text search and shall not appear in a response record. Predicates that include this pseudo-property shall be evaluated against the values of all ebRIM information items for which a CSW mapping exists in Table 6.

7.7 Multilingual support

The rim:LocalizedString element is used to provide a language-specific property value as indicated by the value of the xml:lang attribute; the attribute value shall conform to the syntax specified in RFC 4646, and no sibling rim:LocalizedString elements shall have the same value.

Note The primary language may be qualified with a subtag indicating a regional dialect or writing system variant. The IANA registry of language subtags is available at this URL: <http://www.iana.org/assignments/language-subtag-registry>.

If a request contains the `Accept-Language` header field, only rim:LocalizedString elements that match the preferred language(s) shall be included in the response. If no preference is expressed or if there are no matching localizations, all rim:LocalizedString elements shall be included.

7.8 Spatial references

Two kinds of spatial references may be used to characterize the geographic extent or coverage of a registry object: location identifiers such as place names or country subdivisions, and geographic coordinates. These are expressed as slot values, where the slotType attribute refers to a classification scheme defining a set of location codes or to a classification node in the canonical data type scheme.

Several geometry data types are defined in the Basic extension package; these data types all use GML 3.1 as the base lexical representation. A geometry value shall be associated with a coordinate reference system (CRS) using the srsName attribute. No default reference system is identified in this profile, although the World Geodetic System 1984 (WGS 84) is a global 2D geographic CRS that is widely used. A CRS shall be identified using the URN syntax documented in OGC 06-023r1 [3].

EXAMPLE The URN value "urn:ogc:def:crs:EPSG:4326" denotes the entry for the CRS definition in the EPSG geodetic dataset having the code value 4326 (WGS 84).

7.9 Use of the SOAP messaging framework

7.9.1 HTTP binding

An implementation may support the W3C SOAP 1.2 messaging framework. If so, this shall be advertised in the capabilities document by including the value "application/soap+xml" for the operation-specific `Content-Type` constraint. A service that supports SOAP shall conform to the SOAP 1.2 HTTP binding (SOAP Part 2); such a binding shall also be declared in a WSDL service description if one is available. A WSDL 2.0 interface description that includes SOAP bindings is listed in Annex C.2.

The HTTP binding employs two message exchange patterns: "Request-Response" and "SOAP Response". In general, a request bound to the GET method is restricted to the "SOAP response" pattern; POST requests are restricted to the "Request-Response pattern. A conforming implementation shall support both patterns. If a request message does not contain a SOAP envelope—as in the GET method—but a SOAP response is desired, the `Accept` request header shall be used to indicate this preference. The normal response is produced if the service cannot function as a SOAP node.

EXAMPLE Accept: application/soap+xml; application/xml; q=0.8

Request and response messages that contain a SOAP message construct shall satisfy all requirements in section 5 in the W3C SOAP 1.2 specification. The SOAP body element shall contain the appropriate XML request or response entity. This specification imposes no constraints on the content of the optional SOAP header element; it may be ignored if present.

The action parameter in the Content-Type request header field value, if present, may be used to optimize message dispatching and routing [RFC 3902]. The value should include the following service type identifier: "urn:ogc:serviceType:WebRegistryService:1.0".

7.9.2 Fault handling

In the event that an exception report is produced for any reason, a single SOAP Fault element information item shall be included as the only child element information item of the SOAP body element (see SOAP12, sec. 5.4). The elements of the SOAP Fault are constructed as follows:

a) env:Code/env:Value = "env:Sender" or "env:Receiver", depending on the source of the error and the HTTP status code (i.e., 4xx, 5xx);

b) env:Code/env:Subcode/env:Value = value of ows:Exception[1]/@exceptionCode (an OWS exception code value shall be prepended with "ows:");

c) env:Reason/env:Text = ows:Exception[1]/ows:ExceptionText[1], where @xml:lang = ows:ExceptionReport/@language;

d) env:Detail contains the original ows:ExceptionReport element

NOTE The "env" prefix maps to the namespace URI "http://www.w3.org/2003/05/soap-envelope".

7.10 Use of the OGC filter grammar

7.10.1 XPath predicates in OGC filter elements

The OGC filter grammar employs XPath expressions as a means of addressing parts of an XML information set. It restricts the use of XPath predicates in csw:PropertyName elements. A predicate may contain only an abbreviated equality expression that includes a call to the position() function so as to constrain the context position when referring to multi-valued properties.

Example 1 Use of abbreviated XPath syntax to test context position.

```
rim:InternationalString/rim:LocalizedString[2] is equivalent to
rim:InternationalString/rim:LocalizedString[position()=2]
```

This profile relaxes the aforementioned constraint and also permits the use of relative location paths in order to constrain the value of a slot attribute.

Example 2 Use of a relative location path to filter on a slot attribute.

```
rim:Service/rim:Slot[@slotType="urn:ogc:def:dataType:ISO-
19107:GM_Envelope"]/wrs:ValueList/wrs:AnyValue
```

7.10.2 Taxonomy filters

A filter expression encapsulates one or more predicates so as to operate on registry objects that satisfy specified criteria. Such criteria may be applied to query, update, and delete statements. Whenever a filter predicate refers to a branch node in some classification scheme, it shall be automatically expanded to encompass all descendant nodes in that scheme.

7.11 Security considerations

7.11.1 Authentication

Authentication is the process of verifying the identity of a user. It is strongly recommended that implementations employ an authentication mechanism to verify the identity of a user that seeks to modify registry content. The standard HTTP authentication schemes documented in RFC 2617 [1] are widely employed and are suitable for most applications:

a) Basic with TLS (or SSL v3) to establish a secure communication channel;

b) Digest.

In the event that a request requires valid authentication credentials and these are either missing or invalid, the response status code shall be 401 (Unauthorized).

7.11.2 Access control

Authorization is the process of deciding whether or not to permit some action on a registry object. It is expected that anonymous users are permitted to retrieve content from the registry without restriction. However, any request to modify registry content should be subject to authorization. If an access control facility is available, the following default access control policy shall be enforced by a conforming implementation:

a) Only the submitter of a registry object may update or delete it.

b) All users may retrieve a representation of any registry object.

No authorization mechanisms or other access control policies are prescribed in this profile in order to ensure that only authorized users can perform actions on registry objects. However, ebRIM does mandate the use of XACML (the OASIS eXtensible Access Control Markup Language) to specify fine-grained access control policies.

Service providers are free to define and enforce more complex access control policies at any level of granularity: service, request, or registry object. In the event that a requester lacks sufficient privileges to perform some action, the response status code shall be 403 (Forbidden).

8 GetCapabilities

8.1 Introduction

The mandatory GetCapabilities request 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 characteristics of the service; the description is structured into subsections that may be retrieved individually by name.

8.2 GetCapabilities request

The GetCapabilities request is specified in cl. 7.2 of OGC 05-008 (OWS Common 1.0) and cl. 10.5 of OGC 07-006r1. The GET method shall be supported. Support for the POST method is optional; the allowed content encoding(s) (“application/xml” and/or “application/x-www-form-urlencoded”) shall be advertised in the capabilities document as values of the operation-specific “Content-Type” constraint.

The value of the *service* parameter shall be the service type code “CSW-ebRIM”. The value of the *version* parameter shall be “1.0.0”.

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 set of recognized section names shall

be as specified in Table 7 of OGC 05-008 and Table 58 of OGC 07-006r1, with the exception of "Contents".

8.3 GetCapabilities response

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

- [local name] = "Capabilities"
- [namespace name] = "http://www.opengis.net/cat/wrs/1.0"

A sample capabilities document is listed in Annex C.1. The corresponding type definition is shown in the code listing below. See clauses 10.5 and 10.13 in OGC 07-006r1 for details.

```
<xsd:element name="Capabilities" type="csw:CapabilitiesType" />
```

8.4 Extended capabilities

Some additional service metadata information items are introduced in this profile; they are contained in the ows:ExtendedCapabilities element or as general constraints in the ows:OperationsMetadata element, and may appear in any order:

a) The wsdi:wsdlLocation attribute includes a list of pairs of IRIs where the first is an absolute IRI that indicates a WSDL 2.0 (or 1.1) namespace name, and the second is a hint for locating a WSDL 2.0 (or 1.1) document that describes the service [5].http://www.w3.org/TR/wsdl20/ - Location-1093-summary

b) The global "srsName" constraint element contains a sequence of absolute URI values that identify supported coordinate reference systems for geometry values; a regular expression may be used to indicate a set of reference systems defined by the same authority.

Example Extended capabilities for CSW-ebRIM.

```
<ows:Constraint name="srsName">
  <ows:Value>urn:ogc:def:crs:EPSG:\d{4,5}</ows:Value>
  <ows:Metadata xlink:type="simple"
    xlink:title="EPSG geodetic parameters"
    xlink:href="http://www.epsg-registry.org/" />
</ows:Constraint>

<ows:ExtendedCapabilities
  wsdi:wsdlLocation="http://wrs1.domain.tld/1.0/wsdl
  http://host.domain.tld/wrs/wsdl" />
```


9 DescribeRecord

9.1 Introduction

The mandatory DescribeRecord request is specified in clause 10.6 of OGC 07-006r1. This request allows a client to discover the information model(s) supported by the catalogue and to retrieve record type definitions.

9.2 DescribeRecord request

The XML representation of the entity body in a request submitted using the POST method shall conform to the csw:DescribeRecord element declaration.

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

The outputFormat attribute may be used to specify an alternative format for the response; this may be requested, for example, to obtain a more human readable representation. Supported values for this parameter shall be indicated in the capabilities document using an operation-specific ows:Parameter element.

9.3 DescribeRecord response

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

- [local name] = "DescribeRecordResponse".
- [namespace name] = "http://www.opengis.net/cat/csw/2.0.2"

If there were no matching schema components, the document element shall be empty.

The content of a csw:SchemaComponent element may be a complete schema or a schema component. If the latter, the value of the parentSchema attribute shall be a URI reference to the source schema.

10 GetRecords

10.1 Introduction

The mandatory GetRecords request is described in clause 10.8 of OGC 07-006r1. This is the principal request used to search and retrieve catalogue content. Some, all, or none of the registry objects in the result set may be included in the response message. Asynchronous or distributed query processing is beyond the scope of this profile.

10.2 GetRecords request

10.2.1 POST method

The XML representation of the message body in a request submitted using the POST method shall be a valid csw:GetRecords element. The csw:ResponseHandler and csw:DistributedSearch elements, if present, shall be ignored and the request processed in the normal synchronous manner by the recipient.

A csw:GetRecords element must include a query statement. The following query elements shall be supported by a conforming implementation:

a) A csw:Query element as specified in clause 10.8.4 of OGC 07-006r1. The OGC filter grammar (OGC 04-095) shall be supported by all conforming implementations. See clause 10.2.3 for guidance about using the generic syntax to query ebRIM objects.

b) A rim:AdhocQuery element as specified in section 7.3 of ebRIM 3.0. Such an element may invoke a stored query (clause 16) or specify a query to execute; if the latter, the queryLanguage attribute shall identify a supported query language by referring to a node in the canonical query language scheme.

10.2.2 GET method

The request parameters are specified in clause 10.8.2 of OGC 07-006r1. The `ResponseHandler` and `DistributedSearch` parameters shall be ignored.

10.2.3 Applying the CSW query syntax to ebRIM

In order to adapt the generic CSW query syntax to a complex information model such as ebRIM, some additional constraints are required. The declaration of binding variables—or type name 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 csw:Query/@typeNames attribute is a whitespace-separated list of QName values that identifies the registry object types that are in the query scope. One or more aliases may be bound to a type name, where each variable ranges over instances of a given object type. Multiple variables may be bound to a single type; these are delimited by an underscore (‘_’) character.

EXAMPLE 1 Declaring binding variables (type name aliases) in a query.

```
<csw:Query typeNames="rim:Service_srv1_srv2 wrs:ExtrinsicObject">
  <!-- query specification -->
</csw:Query>
```

Within a query filter, a binding variable is referenced as shown below, where NCName matches a declared type name alias:

```
VariableReference ::= '$' NCName
```


EXAMPLE 2 Query with binding variables

```
<csw:Query typeNames='wrs:ExtrinsicObject rim:Association_a1'>
 <csw:ElementSetName
    typeNames='wrs:ExtrinsicObject'>full</ElementSetName>
 <csw:Constraint version='1.1.0'>
  <ogc:Filter>
   <ogc:And>
    <ogc:PropertyIsEqualTo>
     <ogc:PropertyName>$a1/@associationType</ogc:PropertyName>
     <ogc:Literal>
     urn:ogc:def:ebRIM-AssociationType:OGC:OperatesOn
     </ogc:Literal>
    </ogc:PropertyIsEqualTo>
    <ogc:PropertyIsEqualTo>
     <ogc:PropertyName>$a1/@sourceObject</ogc:PropertyName>
     <ogc:Literal>
     urn:uuid:86084c6a-292f-4687-bf52-aef49b5ff2d6
     </ogc:Literal>
    </ogc:PropertyIsEqualTo>
    <ogc:PropertyIsEqualTo>
     <ogc:PropertyName>wrs:ExtrinsicObject/@id</ogc:PropertyName>
     <ogc:PropertyName>$a1/@targetObject</ogc:PropertyName>
    </ogc:PropertyIsEqualTo>
    <ogc:PropertyIsEqualTo>
     <ogc:PropertyName>
     wrs:ExtrinsicObject/@objectType
     </ogc:PropertyName>
     <ogc:Literal>
     urn:ogc:def:ebRIM-ObjectType:OGC:Dataset
     </ogc:Literal>
    </ogc:PropertyIsEqualTo>
   </ogc:And>
  </ogc:Filter>
 </csw:Constraint>
</csw:Query>
```

10.3 GetRecords response

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

- [local name] = “GetRecordsResponse”.
- [namespace name] = “http://www.opengis.net/cat/csw/2.0.2”.

The csw:RequestId element shall be included only if the client supplied a value in the original request message. The search results, if present, shall include a sequence of either rim:RegistryObject elements or elements that substitute for csw:AbstractRecord, depending on the value of the outputSchema request parameter (Table 7). In any case elements belonging to the appropriate substitution group may appear, where these typically correspond to different views or instances of subtypes.

Table 7 — Representations corresponding to known output schemas

Value of outputSchema attribute	Record representation(s)
http://www.opengis.net/cat/csw/2.0.2	csw:Record csw:SummaryRecord csw:BriefRecord
urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0	rim:RegistryObject[a]
a Representations of registry object subtypes are included if a "full" view is requested.	

If the resultType attribute in the request has the value "validate" (i.e., process the request asynchronously), the response shall include an exception with code "wrs:NotSupported".

If a supplied query is expressed in an unsupported query language, the response shall include an exception with code "wrs:NotSupported".

11 GetRecordById

11.1 Introduction

The mandatory GetRecordById request is specified in clause 10.9 of OGC 07-006r1. This request provides a simple means of retrieving one or more registry objects by identifier. The supplied identifier values are checked for matches against either the id attribute of a registry object (rim:RegistryObject/@id) or an external identifier (rim:ExternalIdentifier/@value) assigned to a registry object.

11.2 GetRecordById request

The XML representation of the message body in a request submitted using the POST method shall be a valid csw:GetRecordById element.

11.3 GetRecordById response

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

- [local name] = "GetRecordByIdResponse".
- [namespace name] = "http://www.opengis.net/cat/csw/2.0.2".

The document element shall contain registry object representations corresponding to the requested output schema and element set (view). A registry object in the result set shall satisfy any of the following cases:

a) have a matching @id attribute value; or

b) have a child rim:ExternalIdentifier element with a matching @value attribute.

12 GetRepositoryItem

12.1 Introduction

The mandatory GetRepositoryItem request 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 shall be an instance of the media type indicated by the value of the Content-Type entity header field.

12.2 GetRepositoryItem request

This request is submitted using the GET method. The values of the *request* and *service* parameters shall be "GetRepositoryItem" and "CSW-ebRIM", respectively. The mandatory *id* parameter shall be an absolute URI that specifies the identifier of the extrinsic object that describes the item.

12.3 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 with status code 200. The `Content-Type` header shall correctly identify the media type of the enclosed resource. If an additional encoding has been applied to the resource—such as compression using gzip—this shall be indicated by the `Content-Encoding` entity header.

If the supplied identifier does not match any registry object or if a repository item cannot be located, the response shall include a status code of 404.

13 GetDomain

13.1 Introduction

The optional GetDomain request is specified in clause 10.7 of OGC 07-006r1. This request 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. A possible use of this request is to discover 'active' terms in a taxonomy that are currently being 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.

13.2 GetDomain request

The XML representation of the entity body in a request submitted using the POST method shall be a csw:GetDomain element. The value of the csw:PropertyName element shall be an XPath expression (location path) that refers to some ebRIM property. Many of the ebRIM properties can be accessed in this manner, and those that refer to nodes in canonical classification schemes can be queried in this manner.

As a special case, a request for a list of published—rather than supported—object types must identify only those object types of which instances 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. In general, all properties that refer to nodes in canonical classification schemes shall be handled in this manner.

EXAMPLE Request a listing of all catalogued object types.

```
<csw:GetDomain service="CSW-ebRIM" version="1.0.0">
  <csw:PropertyName>
  rim:RegistryObject/@objectType
  </csw:PropertyName>
</csw:GetDomain>
```

13.3 GetDomain response

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

- [local name] = “GetDomainResponse”.
- [namespace name] = “http://www.opengis.net/cat/csw/2.0.2”

If the request specifies an unknown parameter or property, the response shall include a status code of 404; an exception with the code wrs:NotFound may also be included in the body of the response.

14 Harvest

14.1 Introduction

The optional Harvest request is specified in clause 10.12 of OGC 07-006r1. This request allows a user to request that the registry attempt to harvest a resource from a specified network location, thereby realizing a 'pull' model for publishing registry content. If the catalogue successfully retrieves the resource and successfully ingests it, then a root registry object plus zero or more related registry objects are created or updated. Brief representations of all modified records are returned to the client when processing is complete.

14.2 Harvest request

The XML representation of the message body shall be a valid csw:Harvest element. The csw:ResponseHandler element, if present, shall be ignored and the request processed in a synchronous manner by the recipient.

The “http” URI scheme must be supported by all conforming implementations. All supported URI schemes shall be advertised in the service capabilities document using the “uri-schemes” constraint for the Harvest operation; the values are scheme names from the IANA registry.

Note The IANA registry of URI schemes is available at <http://www.iana.org/assignments/uri-schemes.html>.

The value of the csw:ResourceType element shall be an absolute URI that identifies the kind of resource to harvest. Some common metadata resource types are listed in Table 69 of OGC 07-006r1. Additional resource types shall be defined in an extension package, and metadata extraction rules may prescribe what registry objects shall be created on harvest. Supported resource types shall be advertised in the service capabilities document as values of the "ResourceType" parameter for the Harvest operation.

If present, the value of the csw:ResourceFormat element shall indicate the media type of the resource to harvest. Supported media types are advertised in the service capabilities document using a global "media-types" constraint.

14.3 Harvest response

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

- [local name] = "HarvestResponse"
- [namespace name] = "http://www.opengis.net/cat/csw/2.0.2"

The document element must contain a csw:TransactionResponse element that includes the csw:InsertResults element; this element shall list all registry objects that were created as a result of the harvesting operation. There must be at least one entry in the list that identifies the source extrinsic object; the application of metadata extraction rules may generate additional derived registry objects.

If the format (media type) of the resource is not supported by the catalogue or if the resource type is not recognized, an exception with code wrs:NotSupported shall be returned. In the event that request processing cannot be completed for any reason, an exception with code wrs:TransactionFailed shall be returned.

15 Transaction

15.1 Introduction

The optional Transaction request is specified in clause 10.11 of OGC 07-006r1. This request allows a user to insert, update, or delete registry objects. A transaction request constitutes an atomic change set: all subsidiary actions must be successfully executed or the entire transaction fails. The provision of lifecycle management, auditing, and versioning facilities are beyond the scope of this specification.

15.2 Transaction request

15.2.1 General behaviour

The XML representation of the message body shall be a valid csw:Transaction element. Representations of a registry object shall be contained in csw:Insert or csw:Update elements wherever the wildcard element (xsd:any) appears in the schema type definition. No other elements may appear in these contexts.

Mechanisms for managing the lifecycle of registry content is beyond of scope of this specification. No state transitions are defined, and in the absence of any kind of lifecycle management facility the status attribute of all registry objects shall have the value "urn:oasis:names:tc:ebxml-regrep:StatusType:Submitted". Such a facility, if provided, shall comply with the ebRIM 3.0 specification.

This profile does not address the creation and maintainenance an audit trail—a sequence of rim:AuditableEvent objects that record who modified what objects and when.

This profile does not specify any form of versioning. The value of the versionName attribute of the rim:VersionInfo element shall be an empty (zero-length) string for all unversioned registry objects.

15.2.2 Insert statements

A csw:Insert element shall include one or more registry object (but **not** rim:RegistryObject) representations.

When inserting repository items, the multipart/related content type [RFC 2387] shall 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. An XML document with the csw:Transaction element (or a SOAP envelope) as the root element shall be included in the body of the root part. A repository item is included in a message part, with the `Content-Type` and `Content-ID` headers of that part set as shown in Figure 5.

```
Content-Type: multipart/related;type="application/xml"
```

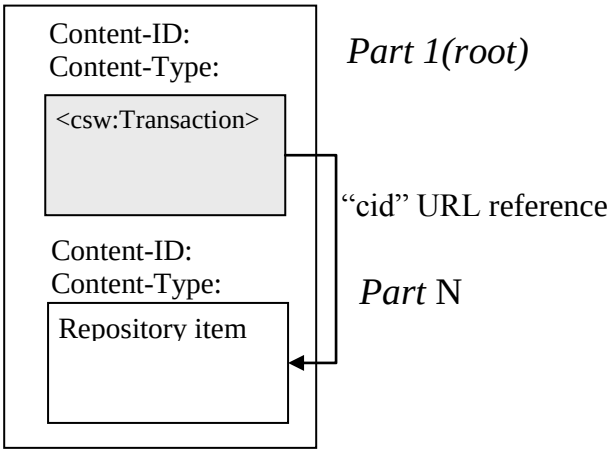

Figure 5 – A multipart request message

The Content-ID header field shall be used to uniquely identify MIME entities in each message part. For each extrinsic object in the root part (for which a repository item exists), the wrs:repositoryItemRef element shall have an xlink:href attribute, the value of which is a URL conforming to the 'cid' scheme. The URL value refers to a specific body part of a message as described in RFC 2392.

Example Using a 'cid' URL to refer to a part having Content-ID = <132d2g457af3@domain2.domain1>.


```
<wrs:ExtrinsicObject id="eo-1" mimeType="application/xml">
  <wrs:repositoryItemRef xlink:type="simple"
    xlink:href="cid:132d2g457af3@domain2.domain1" />
</wrs:ExtrinsicObject>
```

Note A 'cid' URL value has the form of a URL-encoded `addr-spec` from RFC 822.

The XML-binary Optimized Packaging (XOP) convention may be used as an alternative packaging format for assembling multipart requests. Although XOP constructs are generally used in SOAP bindings, the use of SOAP is not required. A multipart XOP package shall be created and processed as specified in the W3C XOP specification. A conforming implementation shall support multipart requests constructed in this manner.

A XOP package shall be contained within a multipart/related message structure, and the Content-Type header of the root part shall have the value "application/xop+xml". The href attribute of the xop:Include element shall be a valid URI in accord with the 'cid' URL scheme.

Example Using a 'cid' URL to refer to a repository item in a XOP package.

```
<wrs:ExtrinsicObject id="eo-1" mimeType="application/xml">
  <wrs:repositoryItem>
    <xop:Include href="cid:132d2g457af3@domain2.domain1" />
  </wrs:repositoryItem>
</wrs:ExtrinsicObject>
```

15.2.3 Update statements

If a representation of a registry object is included in an update statement, this shall either replace the existing object or create a new version if it is a versioned object.

When specifying a partial update that modifies selected properties of one or more registry objects, the value of the csw:RecordProperty/csw:Name element shall be an XPath location path that identifies the property to be updated. If a csw:Value element is not supplied, the property shall be removed if doing so does not violate a schema constraint; otherwise an exception is raised with code wrs:InvalidRequest.

Slot may be selectively updated using a rim:ObjectRef element that identifies the parent registry object to be modified. Slots values may added, replaced, or removed in this manner. If a slot of the same name already exists, the slot values are replaced. A slot is deleted by supplying an empty rim:ValueList element.

EXAMPLE Adding and removing a slot.

```
<csw:Update>
  <rim:ObjectRef
    id="urn:uuid:c8409773-177f-4266-bac9-fa98f73b5664">
    <rim:Slot name="Coverage"
      slotType="urn:ogc:def:ClassificationScheme:ISO-3166-
1:2006:Countries">
      <rim:ValueList>
         <rim:Value>CA</rim:Value>
```



```
      </rim:ValueList>
    </rim:Slot>
    <rim:Slot name="Rights">
      <rim:ValueList />
    </rim:Slot>
  </rim:ObjectRef>
</csw:Update>
```

15.2.4 Delete statements

A registry object shall be deleted only if there are no references to that object. Whenever an extrinsic object is deleted, the corresponding repository item is also deleted (if it exists).

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.

15.3 Transaction response

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

- [local name] = “TransactionResponse”
- [namespace name] = “http://www.opengis.net/cat/csw/2.0.2”

If a verbose response was requested, the csw:InsertResult element shall be included; it shall contain a csw:BriefRecord element for every registry object created as a result of processing the transaction request. Each csw:BriefRecord element shall contain a dc:type element, the value of which indicates the object type of the corresponding registry object (see CSW mappings in Table 6).

In the event that request processing cannot be completed for any reason, an exception with code wrs:TransactionFailed shall be returned.

16 Stored queries

16.1 Invoking stored queries

Extension packages may define various stored queries in order to provide a simple means of searching a registry. A stored query is essentially a parameterized, named query definition known to the hosting service. The query response shall include a csw:GetRecordsResponse element in the message body. A service provider may restrict which users can execute a given stored query.

Using the POST method with a Content-Type of “application/xml”, a <rim:AdhocQuery> element 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.

Example Invoking a stored query in a GetRecords context (POST method).

```
<csw:GetRecords service="CSW-ebRIM" version="1.0.0"
  resultType="results"
  outputSchema="http://www.opengis.net/cat/wrs/1.0">
  <rim:AdhocQuery id="urn:ogc:def:ebRIM-Query:findServices">
    <rim:Slot name="serviceType"
      slotType="urn:oasis:names:tc:ebxml-regrep:DataType:ObjectRef">
      <rim:ValueList><rim:Value>WFS</rim:Value></rim:ValueList>
    </rim:Slot>
  </rim:AdhocQuery>
</csw:GetRecords>
```

A KVP-style syntax (“application/x-www-form-urlencoded” format) is used to invoke a stored query using the GET method. A query string is appended to the request URI as follows: the value of the mandatory *qid* parameter specifies the stored query by identifier; additional parameters are appended as shown in the following example.

Example Invoking a stored query using the GET method.

```
http://host:port/path?request=Query&service=CSW-ebRIM&qid=urn:ogc:def:
ebRIM-Query:findServices
```

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 with status code 400.

The following optional parameters are implicitly assumed to be defined for any stored query:

- *elementSetName* : Desired view—see Table 5 (default value is “brief”)
- *startPosition* : See OGC 07-006r1, cl. 10.8.4.6
- *maxRecords* : See OGC 07-006r1, cl. 10.8.4.7

16.2 Defining stored queries

A stored query is defined by a rim:AdhocQuery instance that may be managed like any other registry object. The query must be expressed in a supported query language. That is, the value of the rim:QueryExpression/@queryLanguage attribute shall 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* variable references. Within a definition, the slotType attribute shall specify the appropriate type or value domain for the actual parameters. The rim:ValueList element shall be empty in a query definition, unless a default value is declared.

17 Extension packages

17.1 Introduction

The CSW-ebRIM profile is intended to provide a flexible, general-purpose registry service that can be tailored for specific purposes and 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 take advantage of the extensibility points offered by ebRIM; these extensibility points encompass:

- new types of extrinsic objects and external links;
- new kinds of associations that link registry objects;
- additional classification schemes—or classification nodes that augment an existing scheme—for classifying registry content;
- predefined queries that reflect commonly executed search patterns;
- additional slots to further characterize particular types of registry objects.

For example, a ‘Portrayal’ 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.

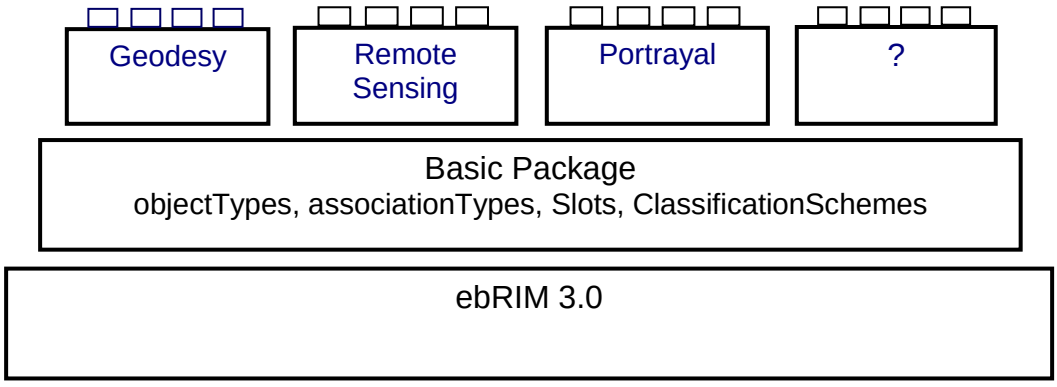

Figure 6 -- Extension packages

An extension package is essentially a container for metadata resources used to describe or characterize other registry objects. It is represented as a rim:RegistryPackage instance. Package members are registry objects that are subject to the following deletion 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. A supplementary specification document shall also be included; it provides general guidance about using the package, defines slots, and stipulates additional constraints that must be satisfied.

Each extension package is itself the member of a 'root' package that contains all packages supported by the service. The identifier of this root package is: "urn:ogc:def:ebRIM-RegistryPackage:OGC:Root".

The **Basic** extension package specification (OGC 07-144r1) is a companion to the CSW-ebRIM application profile—all conforming implementations shall support it. Additional packages may be defined by other parties as needed; these may employ elements of ebRIM and the Basic package as suggested in Figure 6. A package dependency shall be indicated using an association of the following type: "urn:oasis:names:tc:ebxml-regrep:AssociationType:Uses".

17.2 Package content

An extension package is represented as a rim:RegistryPackage item containing elements that may be used to characterize catalogued items. A package may contain the following elements:

a) rim:ClassificationScheme – Represents a controlled vocabulary used to classify registry objects in some manner (e.g., taxonomy, thesaurus).

b) rim:ClassificationNode – Represents a term or concept that augments an existing canonical classification scheme to define a new object type, association type, data type, and so forth.

c) rim:AdhocQuery – Represents a parameterized query definition that provides a simple means of searching registry content without requiring detailed knowledge of ebRIM or a query language.

d) rim:ExtrinsicObject – Represents a repository item that provides a supporting resource such as a specification document or an application schema.

Package authors are strongly encouraged to regard the Basic package as a template for documenting more specialized extension packages.

17.3 Metadata extraction rules

In many cases it may be desirable to ingest a repository item and create a set of registry objects that describe it, thereby promoting some of its content into ebRIM elements. A derived registry object that is created as a result of applying metadata extraction rules to a repository item shall be related to the extrinsic object that describes the resource from which it is derived. The sourceObject attribute identifies the *derived* registry object; the targetObject attribute identifies the *source* extrinsic object. The value of the associationType attribute shall be "urn:ogc:def: ebRIM-AssociationType: OGC: Source" (see OGC 07-144r1, 8.2.8).

In order to ensure that the content of a derived registry object is consistent with the source resource and does not contravene the generation rules that spawned it, the service must prevent any attempt to modify a derived object directly. Subsequent modifications to the source resource should be appropriately propagated to derived registry objects in accord with the applicable metadata extraction rules.

Annex A
(normative)

Abstract test suite

A.1 Test module for conformance level 1

A.1.1 Conformance level 1

a) Purpose: Check that the IUT satisfies all assertions for conformance level 1.

b) Method: Functional testing performed in an automated and/or manual manner. Assess the observable behaviour of the IUT with respect to the following service capabilities:

- GetCapabilities
- DescribeRecord
- GetRecords
- GetRecordById
- GetRepositoryItem
- GetDomain
- Stored queries

c) Reference: OGC 07-110r2

d) Test type: Capability

A.1.2 Test case for GetCapabilities using GET method

Table A.1 – GetCapabilities using GET method

Identifier	http://www.opengis.net/cat/wrs/1.0/atc/level-1/GetCapabilities-Get
Test purpose	Check that GetCapabilities is implemented using the GET method.
Test method	Pass if the assertion is satisfied; fail otherwise.
Reference	- OGC 05-008: cl. 7.2.2 - OGC 07-006r1: cl. 10.5.2
Test type	Basic

A.1.3 Test case for GetRecordById using GET method

Table A.2 – GetRecordById using GET method

Identifier	http://www.opengis.net/cat/wrs/1.0/atc/level-1/GetRecordById-Get
Test purpose	Check that GetRecordById is implemented using the GET method.

Test method	Pass if the assertion is satisfied; fail otherwise.
Reference	- OGC 07-006r1: cl. 10.9.2 - OGC 07-110r2: cl. 7.1
Test type	Basic

A.1.4 Test case for DescribeRecord using GET method

Table A.3 – DescribeRecord using GET method

Identifier	http://www.opengis.net/cat/wrs/1.0/atc/level-1/DescribeRecord-Get
Test purpose	Verify that DescribeRecord is implemented using the GET method.
Test method	Pass if the assertion is satisfied; fail otherwise.
Reference	- OGC 07-006r1: cl. 10.6.2 - OGC 07-110r2: cl. 7.1
Test type	Basic

A.1.5 Test case for GetRecords using POST method

Table A.4 – GetRecords using POST method

Identifier	http://www.opengis.net/cat/wrs/1.0/atc/level-1/GetRecords-Post
Test purpose	Verify that GetRecords is implemented using the POST method.
Test method	Pass if the assertion is satisfied; fail otherwise. The body of the request message contains the csw:GetRecords element.
Reference	- OGC 07-006r1: cl. 10.8.3 - OGC 07-110r2: cl. 7.1, 10.2.1
Test type	Basic

A.1.6 Test case for GetRepositoryItem using GET method

Table A.5 – GetRepositoryItem using GET method

Identifier	http://www.opengis.net/cat/wrs/1.0/atc/level-1/GetRepositoryItem-Get
Test purpose	Verify that GetRepositoryItem is implemented using the GET method.
Test method	Pass if the assertion is satisfied; fail otherwise.
Reference	- OGC 07-110r2: cl. 7.1 - OGC 07-110r2: cl. 12

Test type	Basic

A.1.7 Test case for GetDomain using GET method

Table A.6 – GetDomain using GET method

Identifier	http://www.opengis.net/cat/wrs/1.0/atc/level-1/GetDomain-Get
Test purpose	Verify that GetDomain is implemented using the GET method.
Test method	Pass if the assertion is satisfied; fail otherwise. Conditional: only if the GetDomain operation is advertised in the capabilities document.
Reference	- OGC 07-006r1: cl. 10.7.2 - OGC 07-110r2: cl. 13
Test type	Capability

A.2 Test module for conformance level 2

A.2.1 Conformance level 2

a) Purpose: Confirm that the IUT satisfies all applicable assertions for conformance level 1 and conformance level 2.

b) Method: Functional testing performed in an automated and/or manual manner. Assess the observable behaviour of the IUT with respect to the following service capabilities, at least one of which shall be implemented:

- Harvest
- Transaction

c) Reference: OGC 07-110r2

d) Test type: Basic

A.2.2 Test case for availability of a publishing facility

Table A.7 – Availability of a publishing facility

Identifier	http://www.opengis.net/cat/wrs/1.0/atc/level-2/PublishingFacility
Test purpose	Verify that Harvest or Transaction is implemented.
Test method	Pass if the assertion is satisfied; fail otherwise. At least one of these operations must be advertised in the service capabilities document.
Reference	- OGC 07-110r2: cl. 2.3

Test type	Basic

A.2.3 Test case for Harvest using POST method

Table A.8 – Harvest using POST method

Identifier	http://www.opengis.net/cat/wrs/1.0/atc/level-2/Harvest-Post
Test purpose	Verify that Harvest is implemented using the POST method.
Test method	Pass if the assertion is satisfied; fail otherwise. Conditional: only if the Harvest operation is advertised in the capabilities document.
Reference	- OGC 07-006r1: cl. 10.12 - OGC 07-110r2: cl. 14
Test type	Capability

A.2.4 Test case for Transaction using POST method

Table A.9 – Transaction using POST method

Identifier	http://www.opengis.net/cat/wrs/1.0/atc/level-2/Transaction-Post
Test purpose	Verify that Transaction is implemented using the POST method.
Test method	Pass if the assertion is satisfied; fail otherwise. Conditional: only if the Transaction operation is advertised in the capabilities document.
Reference	- OGC 07-006r1: cl. 10.11 - OGC 07-110r2: cl. 15
Test type	Capability

Annex B
(normative)

XML Schema for CSW-ebRIM

```
<xsd:schema id="csw-ebrim"
  targetNamespace="http://www.opengis.net/cat/wrs/1.0"
  xmlns:wrs="http://www.opengis.net/cat/wrs/1.0"
  xmlns:rim="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0"
  xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xmime="http://www.w3.org/2005/05/xmlmime"
  elementFormDefault="qualified"
  version="1.0.0">

  <xsd:annotation>
    <xsd:appinfo xmlns:sch="http://www.ascc.net/xml/schematron">
      <sch:pattern id="ComplexSlotValuesPattern"
name="ComplexSlotValuesPattern">
        <sch:rule context="//wrs:ValueList">
          <sch:report test="rim:Value">rim:Value not allowed in this
context: expected wrs:AnyValue.</sch:report>
        </sch:rule>
      </sch:pattern>
    </xsd:appinfo>
    <xsd:documentation xml:lang="en">
    Schema for CSW-ebRIM catalogue profile (OGC 07-110r2).
    </xsd:documentation>
  </xsd:annotation>

  <xsd:import namespace="http://www.opengis.net/cat/csw/2.0.2"
    schemaLocation="http://schemas.opengis.net/csw/2.0.2/CSW-
publication.xsd" />
  <xsd:import namespace="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0"
    schemaLocation="http://docs.oasis-
open.org/regrep/v3.0/schema/rim.xsd" />
  <xsd:import namespace="http://www.w3.org/1999/xlink"
schemaLocation="http://schemas.opengis.net/xlink/1.0.0/xlinks.xsd" />
  <xsd:import namespace="http://www.opengis.net/ogc"
schemaLocation="http://schemas.opengis.net/filter/1.1.0/filter.xsd"/>

  <xsd:element name="Capabilities" type="csw:CapabilitiesType" />
  <xsd:element name="RecordId" type="wrs:RecordIdType"
    substitutionGroup="ogc:_Id" id="RecordId">
    <xsd:annotation>
    <xsd:documentation xml:lang="en">
    A general record identifier, expressed as an absolute URI
```

```
    that maps to the rim:RegistryObject/@id attribute. It
    substitutes for the ogc:_Id element in an OGC filter
    expression.
    </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:complexType name="RecordIdType" id="RecordIdType">
    <xsd:complexContent mixed="true">
      <xsd:extension base="ogc:AbstractIdType" />
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="ExtrinsicObject" type="wrs:ExtrinsicObjectType"
    substitutionGroup="rim:ExtrinsicObject"/>
  <xsd:complexType name="ExtrinsicObjectType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
      Extends rim:ExtrinsicObjectType to add the following:
      1. MTOM/XOP based attachment support.
      2. XLink based reference to a part in a multipart/related
         message structure.
      NOTE: This is planned for RegRep 4.0.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="rim:ExtrinsicObjectType">
        <xsd:choice minOccurs="0" maxOccurs="1">
          <xsd:element name="repositoryItemRef"
                       type="wrs:SimpleLinkType"/>
          <xsd:element name="repositoryItem"
                       type="xsd:base64Binary"
                       xmime:expectedContentTypes="*/*">

          </xsd:element>
        </xsd:choice>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="ValueList" type="wrs:ValueListType"
    substitutionGroup="rim:ValueList" />
  <xsd:complexType name="ValueListType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">Allows complex slot
values.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="rim:ValueListType">
        <xsd:sequence minOccurs="0" maxOccurs="unbounded">
          <xsd:element ref="wrs:AnyValue" />
        </xsd:sequence>
      </xsd:extension>
```



```
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="AnyValue" type="wrs:AnyValueType"/>
  <xsd:complexType name="AnyValueType" mixed="true">
    <xsd:sequence>
      <xsd:any minOccurs="0" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="SimpleLinkType" id="SimpleLinkType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
      Incorporates the attributes defined for use in simple
      XLink elements.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:attributeGroup ref="xlink:simpleLink" />
  </xsd:complexType>
</xsd:schema>
```

Annex C
(informative)

Example XML documents

C.1 CSW-ebRIM capabilities document

```
<wrs:Capabilities version="1.0.0"
  xmlns:wrs="http://www.opengis.net/cat/wrs/1.0"
  xmlns:ows="http://www.opengis.net/ows"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:xlink="http://www.w3.org/1999/xlink">

  <ows:ServiceIdentification>
    <ows:Title>CSW-ebRIM Registry Service</ows:Title>
    <ows:Abstract>Abstract</ows:Abstract>
    <ows:Keywords>
      <ows:Keyword>registry</ows:Keyword>
    </ows:Keywords>
    <ows:ServiceType codeSpace="http://www.opengeospatial.org/ogcna">
    urn:ogc:serviceType:CatalogueService-ebRIM
    </ows:ServiceType>
    <ows:ServiceTypeVersion>1.0.0</ows:ServiceTypeVersion>
  </ows:ServiceIdentification>

  <ows:ServiceProvider>
    <ows:ProviderName>Alpha Beta, Inc.</ows:ProviderName>
    <ows:ServiceContact>
      <ows:IndividualName>Phineas Fogg</ows:IndividualName>
    </ows:ServiceContact>
  </ows:ServiceProvider>

  <ows:OperationsMetadata>
    <ows:Operation name="GetCapabilities">
      <ows:DCP>
        <ows:HTTP>
          <ows:Get xlink:type="simple"
            xlink:href="http://localhost:80/path" />
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
    <ows:Operation name="GetRecords">
      <ows:DCP>
        <ows:HTTP>
          <ows:Get xlink:type="simple"
            xlink:href="http://localhost:80/path" />
          <ows:Post xlink:type="simple"
            xlink:href="http://localhost:80/path" />
        </ows:HTTP>
```

```
  </ows:DCP>
</ows:Operation>
<ows:Operation name="GetRecordById">
  <ows:DCP>
    <ows:HTTP>
      <ows:Get xlink:type="simple"
        xlink:href="http://localhost:80/path" />
    </ows:HTTP>
  </ows:DCP>
</ows:Operation>
<ows:Operation name="DescribeRecord">
  <ows:DCP>
    <ows:HTTP>
      <ows:Get xlink:type="simple"
        xlink:href="http://localhost:80/path" />
    </ows:HTTP>
  </ows:DCP>
</ows:Operation>
<ows:Operation name="GetDomain">
  <ows:DCP>
    <ows:HTTP>
      <ows:Get xlink:type="simple"
        xlink:href="http://localhost:80/path" />
    </ows:HTTP>
  </ows:DCP>
</ows:Operation>
<ows:Operation name="GetRepositoryItem">
  <ows:DCP>
    <ows:HTTP>
      <ows:Get xlink:type="simple"
        xlink:href="http://localhost:80/path" />
    </ows:HTTP>
  </ows:DCP>
</ows:Operation>
<ows:Operation name="Transaction">
  <ows:DCP>
    <ows:HTTP>
      <ows:Post xlink:type="simple"
        xlink:href="https://localhost:443/path" />
    </ows:HTTP>
  </ows:DCP>
</ows:Operation>
<ows:Operation name="Harvest">
  <ows:DCP>
    <ows:HTTP>
      <ows:Post xlink:type="simple"
        xlink:href="https://localhost:443/path" />
    </ows:HTTP>
  </ows:DCP>
  <ows:Parameter name="ResourceType">
    <ows:Value>http://www.isotc211.org/2005/gmd</ows:Value>
  </ows:Parameter>
  <ows:Constraint name="uri-schemes">
```



```
        <ows:Value>http</ows:Value>
        <ows:Metadata xlink:type="simple" xlink:title="IANA URI
schemes registry"
        xlink:href="http://www.iana.org/assignments/uri-schemes.html"
/>
      </ows:Constraint>
    </ows:Operation>

    <ows:Constraint name="media-types">
      <ows:Value>application/xml</ows:Value>
      <ows:Metadata xlink:type="simple" xlink:title="IANA media types
registry"
        xlink:href="http://www.iana.org/assignments/media-types/" />
    </ows:Constraint>
    <ows:Constraint name="srsName">
      <ows:Value>urn:ogc:def:crs:EPSG:\d{4,5}</ows:Value>
      <ows:Metadata xlink:type="simple"
        xlink:title="EPSG geodetic parameters"
        xlink:href="http://www.epsg-registry.org/" />
    </ows:Constraint>

    <ows:ExtendedCapabilities
      xmlns:wsdi="http://www.w3.org/ns/wsdl-instance"
      wsdi:wsdlLocation="http://wrs1.domain.tld/1.0/wsdl
      http://localhost/wrs/wsdl" />

  </ows:OperationsMetadata>

  <ogc:Filter_Capabilities>
    <ogc:Spatial_Capabilities xmlns:gml="http://www.opengis.net/gml">
      <ogc:GeometryOperands>
        <ogc:GeometryOperand>gml:Envelope</ogc:GeometryOperand>
      </ogc:GeometryOperands>
      <ogc:SpatialOperators>
        <ogc:SpatialOperator name="BBOX" />
      </ogc:SpatialOperators>
    </ogc:Spatial_Capabilities>

    <ogc:Scalar_Capabilities>
      <ogc:LogicalOperators />
      <ogc:ComparisonOperators>
        <ogc:ComparisonOperator>EqualTo</ogc:ComparisonOperator>
<ogc:ComparisonOperator>NotEqualTo</ogc:ComparisonOperator>
        <ogc:ComparisonOperator>LessThan</ogc:ComparisonOperator>
<ogc:ComparisonOperator>GreaterThan</ogc:ComparisonOperator>
<ogc:ComparisonOperator>LessThanEqualTo</ogc:ComparisonOperator>
<ogc:ComparisonOperator>GreaterThanEqualTo</ogc:ComparisonOperator>
        <ogc:ComparisonOperator>Like</ogc:ComparisonOperator>
      </ogc:ComparisonOperators>
    </ogc:Scalar_Capabilities>

    <ogc:Id_Capabilities>
```



```
      <!-- wrs:RecordId substitutes for ogc:_Id in ogc:Filter -->
      <ogc:EID />
    </ogc:Id_Capabilities>
  </ogc:Filter_Capabilities>
</wrs:Capabilities>
```

C.2 WSDL 2.0 interface description

```
<wsd:description
  targetNamespace="http://www.opengis.net/cat/wrs/1.0/wsdl"
  xmlns:tns="http://www.opengis.net/cat/wrs/1.0/wsdl"
  xmlns:wrs="http://www.opengis.net/cat/wrs/1.0"
  xmlns:iri="http://www.opengis.net/cat/wrs/1.0/iri"
  xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
  xmlns:ows="http://www.opengis.net/ows"
  xmlns:wsd="http://www.w3.org/ns/wsdl"
  xmlns:wsdx="http://www.w3.org/ns/wsdl-extensions"
  xmlns:wsoap="http://www.w3.org/ns/wsdl/soap"
  xmlns:whttp="http://www.w3.org/ns/wsdl/http"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <wsd:documentation>
  W3C WSDL interface descriptions for the CSW-ebRIM 1.0 catalogue
service. This document shall be imported by all instance-specific
service descriptions.
  </wsd:documentation>

  <wsd:types>
    <xsd:import
      namespace="http://www.opengis.net/cat/wrs/1.0"
schemaLocation="http://schemas.opengis.net/wrs/1.0.0/wrs.xsd" />
    <xsd:import
      namespace="http://www.opengis.net/cat/wrs/1.0/iri"
      schemaLocation="http://schemas.opengis.net/wrs/1.0.0/wrs-
iri.xsd" />
    <xsd:import
      namespace="http://www.opengis.net/cat/csw/2.0.2"
      schemaLocation="http://schemas.opengis.net/csw/2.0.2/CSW-
publication.xsd" />
    <xsd:import
      namespace="http://www.opengis.net/ows"
schemaLocation="http://schemas.opengis.net/ows/1.0.0/owsAll.xsd" />
  </wsd:types>

  <wsd:interface name="DiscoveryInterface">

    <wsd:fault name="InvalidRequestFault"
      element="ows:ExceptionReport">
      <wsd:documentation>
      The body of the request message is invalid or not well formed.
The response status code is 400 and the OGC exception code is
wrs:InvalidRequest.
      </wsd:documentation>
```



```
    </wsd:fault>

    <wsd:fault name="NotImplementedFault"
      element="ows:ExceptionReport">
      <wsd:documentation>
      Unsupported functionality--the request cannot be processed. The
response status code is 501 and the OGC exception code is
wrs:NotImplemented.
      </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"
      wsdx:safe="true">
      <wsd:documentation>Request OGC service description
(mandatory).</wsd:documentation>
      <wsd:input messageLabel="In" element="iri:GetCapabilities" />
      <wsd:output messageLabel="Out" element="wrs:Capabilities" />
      <wsd:outfault messageLabel="Out" ref="tns:InvalidRequestFault"
/>
    </wsd:operation>

    <wsd:operation name="GetCapabilities-xml"
      pattern="http://www.w3.org/2005/08/wsdl/in-out"
      wsdx:safe="true">
      <wsd:documentation>
      Alternative GetCapabilities request that supplies an XML message
body
      (optional).
      </wsd:documentation>
      <wsd:input messageLabel="In" element="csw:GetCapabilities" />
      <wsd:output messageLabel="Out" element="wrs:Capabilities" />
      <wsd:outfault messageLabel="Out" ref="tns:InvalidRequestFault"
/>
      <wsd:outfault messageLabel="Out" ref="tns:NotImplementedFault"
/>
    </wsd:operation>

    <wsd:operation name="GetRecords-xml"
      pattern="http://www.w3.org/2005/05/wsdl/in-out"
      wsdx:safe="true">
      <wsd:documentation>
      Main search and retrieval facility (mandatory).
      </wsd:documentation>
      <wsd:input messageLabel="In" element="csw:GetRecords" />
      <wsd:output messageLabel="Out" element="csw:GetRecordsResponse"
/>
      <wsd:outfault messageLabel="Out" ref="tns:InvalidRequestFault"
/>
    </wsd:operation>
```



```
    <wsd:operation name="GetRecords"
      pattern="http://www.w3.org/2005/05/wsdl/in-out"
      style="http://www.w3.org/2005/08/wsdl/style/iri"
      wsdx:safe="true">
      <wsd:documentation>
      Alternative GetRecords request using the IRI style (optional).
      </wsd:documentation>
      <wsd:input messageLabel="In" element="iri:GetRecords" />
      <wsd:output messageLabel="Out" element="csw:GetRecordsResponse"
/>
      <wsd:outfault messageLabel="Out" ref="tns:InvalidRequestFault"
/>
      <wsd:outfault messageLabel="Out" ref="tns:NotImplementedFault"
/>
    </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"
      wsdx:safe="true">
      <wsd:documentation>
      Retrieve representations of one or more registry objects by
identifier (mandatory).
      </wsd:documentation>
      <wsd:input messageLabel="In" element="iri:GetRecordById" />
      <wsd:output messageLabel="Out"
element="csw:GetRecordByIdResponse" />
      <wsd:outfault messageLabel="Out" ref="tns:InvalidRequestFault"
/>
    </wsd:operation>

    <wsd:operation name="GetRecordById-xml"
      pattern="http://www.w3.org/2005/08/wsdl/in-out"
      wsdx:safe="true">
      <wsd:documentation>Alternative GetRecordById request that
supplies an XML message body (optional).</wsd:documentation>
      <wsd:input messageLabel="In" element="csw:GetRecordById" />
      <wsd:output messageLabel="Out"
element="csw:GetRecordByIdResponse" />
      <wsd:outfault messageLabel="Out" ref="tns:InvalidRequestFault"
/>
      <wsd:outfault messageLabel="Out" ref="tns:NotImplementedFault"
/>
    </wsd:operation>

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



```
    </wsd:operation>

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

  </wsd:interface>

  <wsd:interface name="InquiryInterface"
extends="tns:DiscoveryInterface">

    <wsd:documentation>
    Includes search and retrieval operations specific to CSW-ebRIM
implementations.
    </wsd:documentation>

    <wsd:operation name="GetRepositoryItem"
      pattern="http://www.w3.org/2005/08/wsdl/in-out"
      style="http://www.w3.org/2005/08/wsdl/style/iri"
      wsdx:safe="true">
      <wsd:documentation>
      Retrieve a repository item described by an ExtrinsicObject
(mandatory).
      </wsd:documentation>
      <wsd:input messageLabel="In" element="iri:GetRepositoryItem" />
      <wsd:output messageLabel="Out" element="#other">
        <wsd:documentation>
        The response body contains the actual repository item.
        </wsd:documentation>
      </wsd:output>
      <wsd:outfault messageLabel="Out" ref="tns:InvalidRequestFault"
/>
    </wsd:operation>

    <wsd:operation name="Query"
      pattern="http://www.w3.org/2005/08/wsdl/in-out"
      style="http://www.w3.org/2005/08/wsdl/style/iri"
      wsdx:safe="true">
      <wsd:documentation>
      Invoke a predefined named query (mandatory).
      </wsd:documentation>
      <wsd:input messageLabel="In" element="iri:Query" />
      <wsd:output messageLabel="Out" element="csw:GetRecordsResponse"
/>
```

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

  </wsd:interface>

  <wsd:interface name="PublishingInterface">

    <wsd:documentation>
    At least one of the publishing operations must be implemented.
    </wsd:documentation>

    <wsd:fault name="InvalidRequestFault"
      element="ows:ExceptionReport">
      <wsd:documentation>
      The body of the request message is invalid or not well formed.
The
      response status code is 400 and the OGC exception code is
wrs:InvalidRequest.
      </wsd:documentation>
    </wsd:fault>

    <wsd:fault name="NotImplementedFault"
      element="ows:ExceptionReport">
      <wsd:documentation>
      Unsupported functionality--the request cannot be processed. The
response
      status code is 501 and the OGC exception code is
wrs:NotImplemented.
      </wsd:documentation>
    </wsd:fault>

    <wsd:fault name="TransactionFailedFault"
      element="ows:ExceptionReport">
      <wsd:documentation>
      The requested transaction could not be completed for some reason
other
      than a validation error. The response status code is 500 and the
OGC
      exception code is wrs:TransactionFailed.
      </wsd:documentation>
    </wsd:fault>

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

      <wsd:input messageLabel="In" element="csw:Transaction"/>
      <wsd:output messageLabel="Out"
element="csw:TransactionResponse"/>
      <wsd:outfault messageLabel="Out" ref="tns:InvalidRequestFault"
/>
      <wsd:outfault messageLabel="Out"
ref="tns:TransactionFailedFault" />
```



```
      <wsd:outfault messageLabel="Out" ref="tns:NotImplementedFault"
/>
    </wsd:operation>

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

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

  <wsd:binding name="Discovery-HttpBinding"
interface="tns:DiscoveryInterface"
    type="http://www.w3.org/ns/wsdl/http">

    <wsd:fault ref="tns:InvalidRequestFault" whttp:code="400" />
    <wsd:fault ref="tns:NotImplementedFault" whttp:code="501" />

    <wsd:operation ref="tns:GetCapabilities" whttp:method="GET" />
    <wsd:operation ref="tns:GetCapabilities-xml" whttp:method="POST"
/>
    <wsd:operation ref="tns:GetRecordById" whttp:method="GET" />
    <wsd:operation ref="tns:GetRecordById-xml" whttp:method="POST" />
    <wsd:operation ref="tns:GetRecords" whttp:method="GET" />
    <wsd:operation ref="tns:GetRecords-xml" whttp:method="POST" />
    <wsd:operation ref="tns:DescribeRecord-xml" whttp:method="POST" />
    <wsd:operation ref="tns:GetDomain-xml" whttp:method="POST" />

  </wsd:binding>

  <wsd:binding name="Inquiry-HttpBinding"
interface="tns:InquiryInterface"
    type="http://www.w3.org/ns/wsdl/http">

    <wsd:fault ref="tns:InvalidRequestFault" whttp:code="400" />
    <wsd:fault ref="tns:NotImplementedFault" whttp:code="501" />

    <wsd:operation ref="tns:GetRepositoryItem" whttp:method="GET"
      whttp:outputSerialization="*/*" />
    <wsd:operation ref="tns:Query" whttp:method="GET" />

  </wsd:binding>

  <wsd:binding name="Publishing-HttpBinding"
interface="tns:PublishingInterface"
```



```
    type="http://www.w3.org/ns/wsdl/http">

    <wsd:fault ref="tns:InvalidRequestFault" whttp:code="400" />
    <wsd:fault ref="tns:NotImplementedFault" whttp:code="501" />
    <wsd:fault ref="tns:TransactionFailedFault" whttp:code="500" />

    <wsd:operation ref="tns:Transaction" whttp:method="POST"

whttp:inputSerialization="application/xml,multipart/related;type='appl
ication/xml'"/>
    <wsd:operation ref="tns:Harvest" whttp:method="POST" />

  </wsd:binding>

  <wsd:binding name="Discovery-SoapBinding"
interface="tns:DiscoveryInterface"
    type="http://www.w3.org/ns/wsdl/soap"
    wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/">

    <wsd:fault ref="tns:InvalidRequestFault"
      wsoap:code="soap:Sender" wsoap:subcodes="wrs:InvalidRequest" />
    <wsd:fault ref="tns:NotImplementedFault"
      wsoap:code="soap:Receiver" wsoap:subcodes="wrs:NotImplemented"
/>

    <wsd:operation ref="tns:GetCapabilities"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>
    <wsd:operation ref="tns:GetCapabilities-xml"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/request-response/"
      wsoap:action="urn:ogc:serviceType:WebRegistryService:1.0" />

    <wsd:operation ref="tns:GetRecordById"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>
    <wsd:operation ref="tns:GetRecordById-xml"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/request-response/"
      wsoap:action="urn:ogc:serviceType:WebRegistryService:1.0" />

    <wsd:operation ref="tns:DescribeRecord-xml"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/request-response/"
      wsoap:action="urn:ogc:serviceType:WebRegistryService:1.0" />
    <wsd:operation ref="tns:GetDomain-xml"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/request-response/"
      wsoap:action="urn:ogc:serviceType:WebRegistryService:1.0" />

    <wsd:operation ref="tns:GetRecords"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>
    <wsd:operation ref="tns:GetRecords-xml"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/request-response/"
      wsoap:action="urn:ogc:serviceType:WebRegistryService:1.0" />
  </wsd:binding>

  <wsd:binding name="Publishing-SoapBinding"
interface="tns:PublishingInterface"
```



```
    type="http://www.w3.org/ns/wsdl/soap"
    wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/">

    <wsd:fault ref="tns:InvalidRequestFault"
      wsoap:code="soap:Sender" wsoap:subcodes="wrs:InvalidRequest" />
    <wsd:fault ref="tns:NotImplementedFault"
      wsoap:code="soap:Receiver" wsoap:subcodes="wrs:NotImplemented"
/>
    <wsd:fault ref="tns:TransactionFailedFault"
      wsoap:code="soap:Receiver"
wsoap:subcodes="wrs:TransactionFailed" />

    <wsd:operation ref="tns:Transaction"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/request-response/"
      wsoap:action="urn:ogc:serviceType:WebRegistryService:1.0">
      <wsd:input>
        <wsoap:module
          ref="http://www.w3.org/2004/08/soap/features/http-
optimization"
          required="true">
          <wsd:documentation>
          The HTTP SOAP Transmission Optimization Feature is required
to handle
          multipart requests containing repository items.
          </wsd:documentation>
        </wsoap:module>
      </wsd:input>
    </wsd:operation>

    <wsd:operation ref="tns:Harvest"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/request-response/"
      wsoap:action="urn:ogc:serviceType:WebRegistryService:1.0" />

  </wsd:binding>
</wsd:description>
```

Bibliography

[1] IETF RFC 2617, *HTTP Authentication: Basic and Digest Access Authentication*. Available from: <http://tools.ietf.org/html/rfc2617>.

[2] IETF RFC 4949, *Internet Security Glossary, Version 2*. Available from: <http://tools.ietf.org/html/rfc4949>.

[3] ISO 11179-6:2005, *Information technology — Metadata registries (MDR) — Part 6: Registration*. Available from: <http://standards.iso.org/ittf/PubliclyAvailableStandards/c035348_ISO_IEC_11179-6_2005(E).zip>.

[4] OGC 06-023r1, *Definition identifier URNs in OGC namespace* (OGC™ Best Practices Paper). Available from: <http://portal.opengeospatial.org/files/?artifact_id=16339>.

[5] W3C WSDL, *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*. Available from: <http://www.w3.org/TR/wsdl20/>.

[6] XACML, *eXtensible Access Control Markup Language (XACML) Version 2.0*. OASIS Standard (2005-02-01). Available from: <http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf>.

