Open Geospatial Consortium Inc.

Date: 2004-09-14

Reference number of this OGC™ document: **OGC 04-017r1**

Version: 0.9.1

Category: OGC™ Implementation Specification

Editor: Richard Martell

OGC™ Catalogue Services – ebRIM (ISO/TS 15000-3) profile of CSW

Copyright notice

Warning

Double Warning: The Catalogue 2.0 Specification is currently in revision to correct a number of known errors. While the changes to the specification will be minimal, the result is that this Application Profile of CSW will also need to be modified. The revisions to the Catalogue Specification should be completed and approved summer of 2005. The results of the Catalogue changes will be released as a Corrigendum. Further, please note that a new version of 04-017r1(this document) is being prepared to reflect the new OASIS ebRIM 3.0 standard as well as the Corrigendum changes to Catalogue. The new version will be posted by May 23rd as OGC document OGC 05-025 **for member consideration.**

Document type:	Discussion Paper
Document subtype:	
Document stage:	Draft
Document language:	English

Contents

i. Preface

The OGC Catalogue Services 2.0 specification (OGC 04-021) establishes a framework for implementing catalogue services that can meet the needs of stakeholders in a wide variety of application domains. This application profile is based on the CSW schemas for web-based catalogues and it complies with the requirements of clause 11 in OGC 04-021.

Suggested additions, changes, and comments regarding this draft report are welcome and encouraged—these may be submitted to the OGC member portal[1] or by email.

ii. Submitting organizations

The following organizations submitted this document to the Open Geospatial Consortium, Inc.

CubeWerx Inc.

Galdos Systems, Inc.

Image Matters, LLC

Ionic Software

Polexis, Inc.

iii. Submission contact points

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

---

[1] <http://portal.opengis.org/>

Contact	Company	Email
J. Davidson	Image Matters, LLC.	johnd @ imagemattersllc.com
J. Lansing	Polexis, Inc.	jeff @ polexis.com
R. Martell	Galdos Systems, Inc.	rmartell @ galdosinc.com
J. Sonnet	Ionic Software	jerome.sonnet @ ionicsoft.com
P. Vretanos	CubeWerx	pvretanos @ cuberwerx.com

iv. Revision history

Date	Release	Editor	Primary clause(s) modified	Description
2004-05-31	0.9.0	R. Martell		Initial draft release.
2004-09-14	0.9.1	R. Martell	front matter (i-vi)	Comply with revised OGC template; new title; updated future work items (vi.); released as a Discussion Paper.

v. Changes to the OGC™ Abstract Specification

The OGC™ Abstract Specification requires/does not require changes to accommodate the technical contents of this document. The needed changes are proposed in OGC documents XX-XXX, XX-XXX, and XX-XXX.

The following is a list of the required changes:

a)

This clause is required in an application profile, whether or not the OGC Catalogue Services Specification requires enhancement. This clause is optional in many other Interoperability Program documents. Note that the specific changes do not need to be specified, only that changes are required and in what areas changes need to be made.

vi. Future work

A number of changes and enhancements have been identified as possible future work items to be addressed before final release as an adopted specification early in 2005. This list is not exhaustive, and some of this work may be deferred or cancelled.

- align with the ebRIM 3.0 specification
- align with the latest/final OWS Common implementation specification
- align with the OGC Filter 1.1 specification
- include formal WSDL (v1.1, v2.0?) interface definitions
- address issues arising from OWS-2 activities (e.g, SOAP bindings, conformance testing)
- address issues arising from implementation experience (i.e. defects, ambiguities)
- develop a companion conformance test suite (under the CITE program)
- include an alternative RDF/XML representation for registry objects (application/rdf+xml, based on RDF Schema)
- add XQuery as an optional query facility
- consider allowing an XUpdate option for fine-grained updates

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.

This document supersedes OGC document 03-024 (*OWS1 Registry Service*). It qualifies as a class 2 profile under ISO 19106 and is dependent upon the following base standards and specifications:

- OpenGIS Catalogue Services Specification 2.0 (OGC 04-021r2)
- Filter Encoding Implementation Specification 1.0 (OGC 02-059)
- IETF RFC 2616 (Hypertext Transfer Protocol -- HTTP/1.1)
- OASIS ebXML Registry Information Model v2.5[2]

Annex G (Design rationale) is informative. All other annexes are normative.

---

[2] The ebXML registry information model is also published as ISO/TS 15000-3.

Introduction

The target audience for this document includes catalogue users, client developers, service implementers, and system testers. The specification encompasses three interrelated views that reflect different viewpoints on a catalogue service. Each viewpoint[3] focuses on different areas of concern:

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

This document defines an application profile of an OGC Catalogue service; it uses the CSW schemas and complies with the rules articulated in clause 11 of the *OpenGIS Catalogue Services Specification*, version 2.0 (OGC 04-021r2). A compliant application profile defines an information model—the conceptual schema—for the information managed by the catalogue service; this model constitutes a public schema that admits one or more data bindings used to represent catalogue content encoded in some data format (Figure 1).

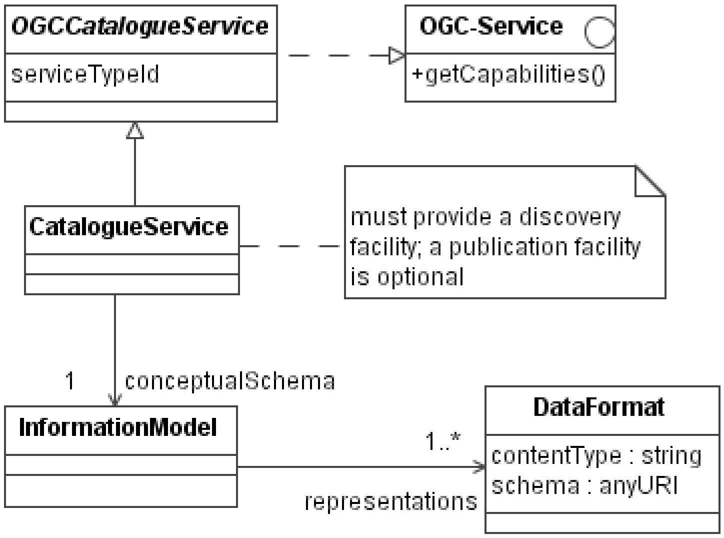

Figure 1 — Catalogue application profiles

---

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

The terms ‘catalogue’ and ‘registry’ are often used interchangeably, but the following distinction is made in this application profile: a registry is a catalogue that exemplifies a formal registration process (e.g., 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 metadata items within some application domain. This profile does not stipulate any registration policies that must be enforced by a conforming implementation. While the WRS profile includes basic facilities to support a registration process, a provider is not required to adopt one; in such cases the service will function as a catalogue.

OGC™ Catalogue Services – ebRIM (ISO/TS 15000-3) profile of CSW

1 Scope

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

2 Conformance

This is a class 2 profile as defined in ISO 19106: it includes extensions permitted within the context of the base specifications and it depends on non-ISO standards.

This conformance clause is required in an OGC Application Profile. Conformance requirements express what a conforming implementation shall do (positively specified requirements), and what it shall not do (negatively specified requirements). These are described in Annex A.

3 Normative references

The following referenced documents are indispensable for the implementation of this profile. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 10746 (all parts), *Information Technology – Open Distributed Processing – Reference Model.*

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

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

IETF RFC 2616, *Hypertext Transfer Protocol -- HTTP/1.1*, Draft Standard (June 1999), available at <http://www.ietf.org/rfc/rfc2026.txt>.

OASIS ebRIM, *ebXML Registry Information Model v2.5*, Committee Approved Specification (June 2003), available at <http://www.oasis-open.org/committees/regrep/documents/2.5/specs/ebrim-2.5.pdf>

OGC 02-094, *Filter Encoding Implementation Specification*, version 1.0 (2001-09-19), available at <http://www.opengis.org/docs/02-059.pdf>

OGC 04-021r2, *OpenGIS Catalogue Services Specification*, version 2.0.

4 Terms and definitions

For the purposes of this document, the following terms and definitions apply /the terms and definitions given in … and the following apply.

4.1

access control policy

set of rules that define the conditions under which access to some resource may occur.

4.2

extrinsic object

a registry object that describes a repository item the content of which is not intrinsic to the information model (e.g., an XML schema, a dataset description).

4.3

registration

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

4.4

registry object

a metadata item published in a registry

4.5

registry service

a catalogue service that adheres to a formal registration procedure.

5 Conventions

5.1 Symbols (and abbreviated terms)

The following abbreviations are used in this application profile:

ACP	Access Control Policy
ebRIM	ebXML Registry Information Model
ISO	International Organization for Standardization
OGC	OpenGIS Consortium
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URN	Uniform Resource Name
WSDL	Web Services Description Language
XACML	Extensible Access Control Markup Language
XML	Extensible Markup Language

5.2 UML Notation

Many of the architecture diagrams that appear in this document are presented as UML class diagrams. The principal UML notations used in this document are summarized in Figure 1. The latest UML specification is the

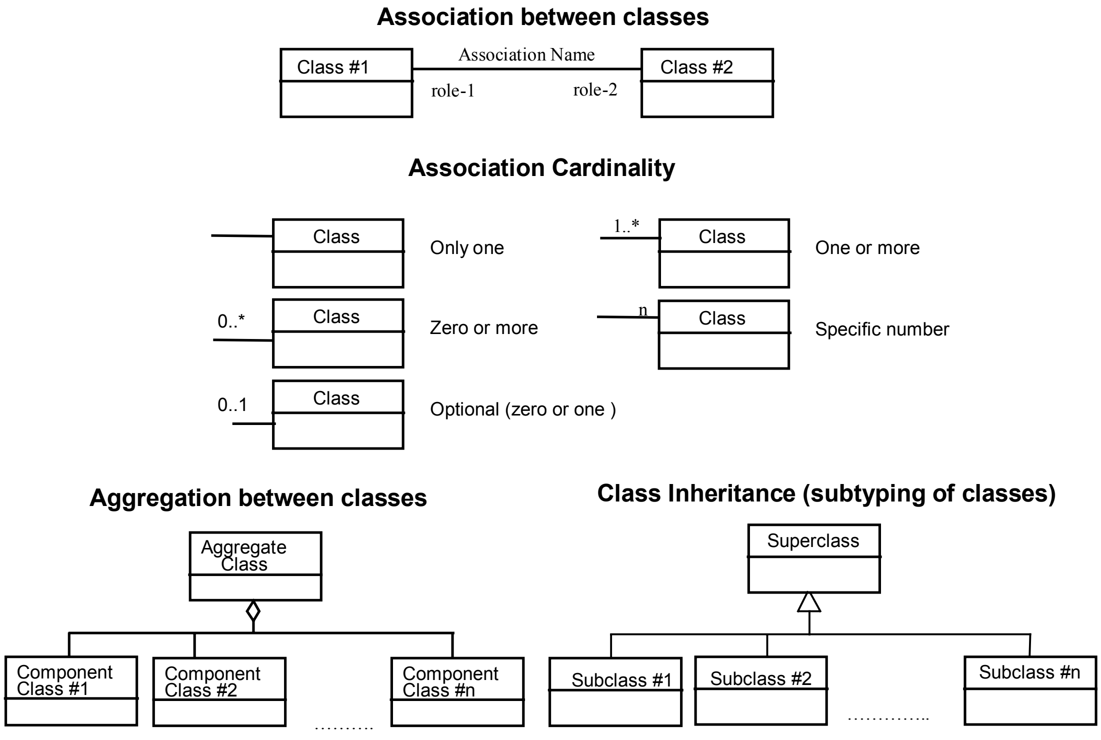

Figure 1 — UML notation

6 System context

6.1 Application domain

A service-oriented architecture must support some fundamental interactions: publishing resource descriptions so that they are accessible to prospective users (*publish*); discovering resources of interest according to some set of search criteria (*discover*); and then interacting with the resource provider to access the desired resources (*bind*). Within such an architecture a catalogue service plays the essential role of matchmaker by providing publication and search functionality, thereby enabling a requester to dynamically discover and communicate with a suitable resource provider (Figure 2).

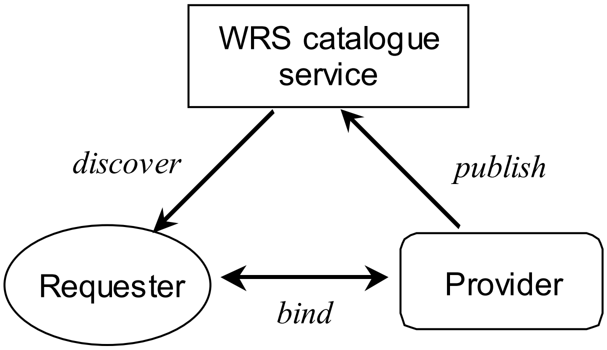

Figure 2 — Essential interactions in a service-oriented architecture

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

A catalogue can function as a stand-alone service or it can interwork with other affiliated catalogues within a federation that spans multiple administrative domains; the federation then effectively enlarges the total *search space* within which resource descriptions may be discovered. When a catalogue is linked to a peer catalogue, it makes the resource descriptions managed by the peer implicitly available to its own clients. Each catalogue client connects to a single catalogue service as its main point of contact with the federation—this is the *agent node*; the propagation of request messages to neighbouring nodes is invisible to the client—it is not necessary to know where the metadata repositories are located or how they are accessed.

The WRS catalogue profile is intended to provide a flexible, general-purpose catalogue service that can be adapted to meet the needs of diverse communities of practice within the geospatial domain.

6.2 Essential use cases

Figure 3 is a context diagram showing the main actors that interact with a WRS catalogue service. An **actor** is someone or something—typically a user or an external system—that is involved in executing a use case.

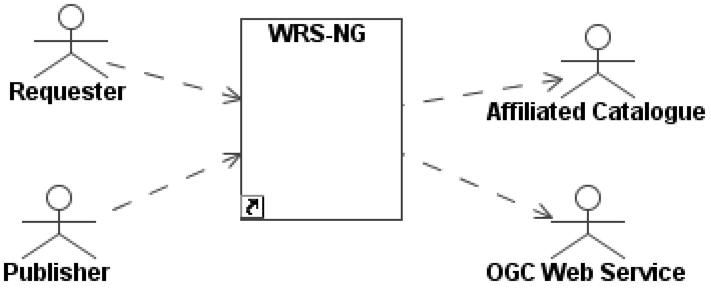

Figure 3 — Main actors

A **use case** describes the sequence of actions that are performed by the system to yield an observable result that is meaningful to a particular actor—it encapsulates a collection of *scenarios*, each of which represents a particular sequence of actions that culminate in either a successful or an unsuccessful outcome. The essential use cases center around the two main catalogue interactions shown in Figure 2: discovery and publication.

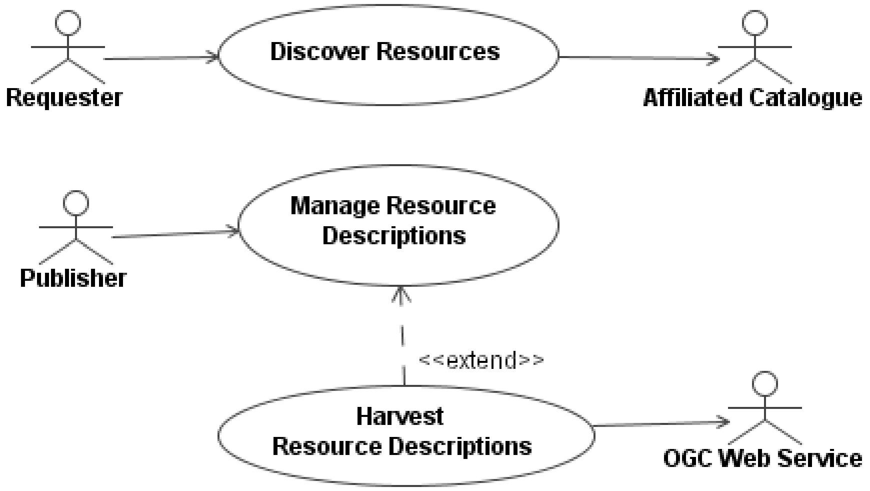

Figure 4 — Principal use cases

The **Discover Resources** use case is presented in Table 1.

Table 1 — Discover Resources	
Actor(s)	Requester, Affiliated Catalogue
Summary	The *Requester* seeks to find resources of interest through simple browsing or by sophisticated query-driven discovery that specifies simple or advanced search criteria. The catalogue performs the search and returns a result set identifier; the result set contains all registry objects that satisfy the search criteria. Alternatively, some or all of the matching registry objects may be returned within the response message. The *Requester* may then choose to retrieve representations of result set items according to some specified schema and element set. If the Requester wants to perform a distributed search and the catalogue is capable of doing so, the request message is forwarded to one or more *Affiliated Catalogues* within a federation of which the WRS catalogue is a member; any 'foreign' results are combined with the local result set.
Trigger(s)	▪ The requester wants to find and retrieve resources of interest

To be completed.

7 Information models

7.1 Capability classes

The WRS profile distinguishes several capability classes that reflect varying degrees of system functionality. These classes are summarized in Table 7.1.

Table 7.1 — Operations provided by WRS capability classes

Capability class label	Operations provided

WRS-BasicDiscovery	OGC-Service.getCapabilities()
	CSW-Discovery.describeRecord()
	CSW-Discovery.getRecords()
	CSW-Discovery.getRecordById()
	CSW-Discovery.getDomain()

WRS-EnhancedDiscovery	*As for WRS-BasicDiscovery, plus*:
	WRS-Retrieval.presentResults()
	WRS-Retrieval.getExtrinsicContent()

Capability class label	Operations provided
WRS-BasicPublication	*Any of the discovery capability classes above, plus*: CSW-Publication.harvest()
WRS-EnhancedPublication	*As for WRS-BasicPublication, plus:* CSW-Publication.transaction()

The `WRS-BasicDiscovery` class is the only mandatory one—it supports synchronous discovery based on the operations provided by the CSW discovery interface. A CSW implementation MUST support at least this level of functionality—all remaining capability classes are optional. Implementing a more capable class implies implementation of less capable classes. For example, a catalogue with a publishing capability must also include a search capability.

The `WRS-EnhancedDiscovery` class adds a couple of additional search and retrieval operations. A user may retrieve items from an existing result set that the user created; the result set may or may not exist within a session context. Content associated with an 'extrinsic' object (see subclause 7.2 for a description of the information model) may be retrieved from some remote data source.

The `WRS-BasicPublication` class adds support for modifying catalogue content though a public interface that allows a 'pull' style of publication. In this case a user may request that the catalogue attempt to harvest a resource from some network location and, if successful, populate the catalogue accordingly.

The `WRS-EnhancedPublication` class adds support for modifying catalogue content though a public interface that allows a 'push' style of publication. A user may insert, update, or delete catalogue entries according to criteria specified in the request message.

The supported capability classes are advertised in the capabilities document as the value of the "http://www.opengis.net/cat/csw/properties/capability-class" service property. The value domain is an enumerated one that includes the labels given in Table 7.1 (there are four permissible values).

7.2 Service information model

7.2.1 OGC service descriptions

Every catalogue service is capable of describing its capabilities to a user in a machine-readable or human-readable format; this capability permits a user to discover how to conduct a conversation with the target service, what kinds of resources it provides, and what additional features are available.

The *OWS Common Implementation Specification* (OGC-04-016) describes a common schema for describing the capabilities of OGC services. Elements of this schema can be mapped to the standard service metadata model specified in ISO 19119, although such a mapping is a partial one.

7.2.2 ISO 19119 service descriptions

The canonical service information model is based on ISO 19119. Figure 7.10 presents the service information model using UML notation. Annex C of ISO 19119 includes the normative data dictionary that defines the service metadata elements appearing in Figure 7.10. Note that some of the metadata elements—those bearing the MD prefix—are imported from ISO 19115; this standard specifies a model for describing geographic datasets.

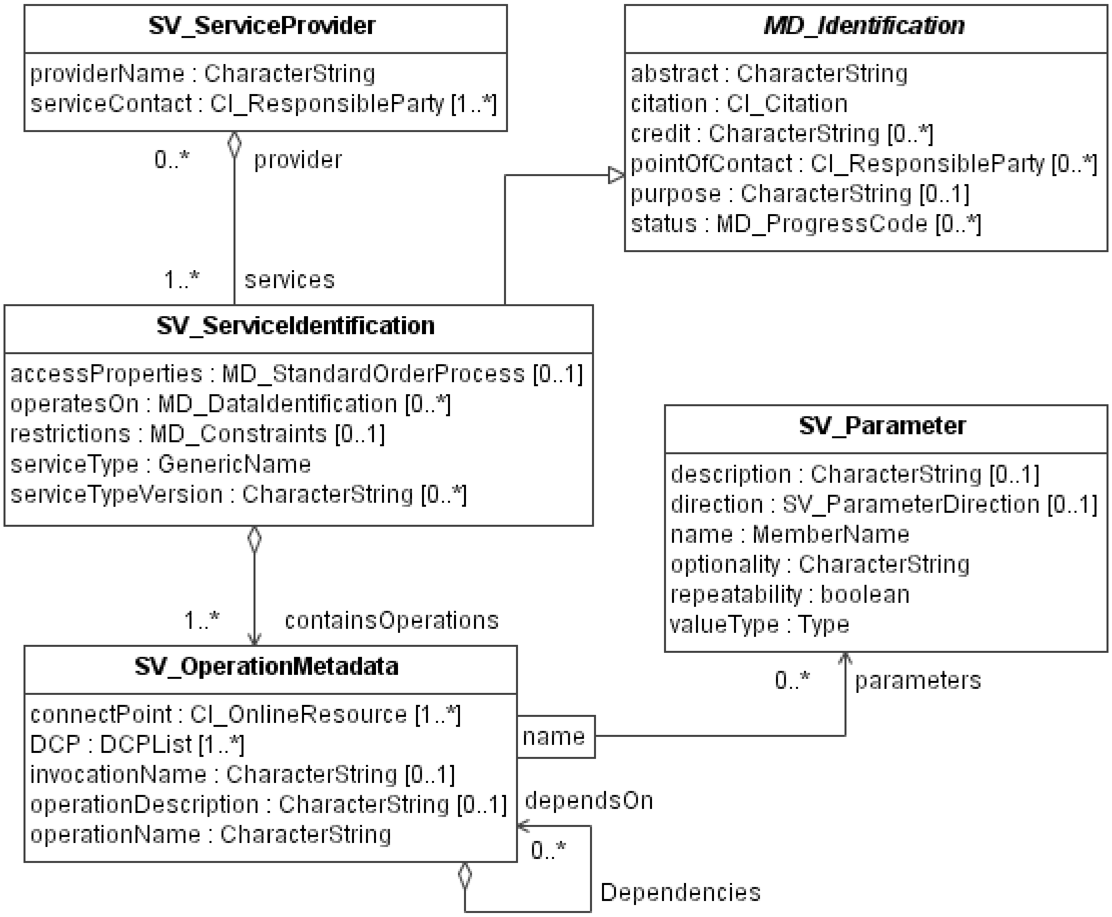
SV_ServiceProvider
providerName : CharacterString
serviceContact : CI_ResponsibleParty [1..*]
0..*
provider
1..*
services
MD_Identification
abstract : CharacterString
citation : CI_Citation
credit : CharacterString [0..*]
pointOfContact : CI_ResponsibleParty [0..*]
purpose : CharacterString [0..1]
status : MD_ProgressCode [0..*]
SV_ServiceIdentification
accessProperties : MD_StandardOrderProcess [0..1]
operatesOn : MD_DataIdentification [0..*]
restrictions : MD_Constraints [0..1]
serviceType : GenericName
serviceTypeVersion : CharacterString [0..*]
SV_Parameter
description : CharacterString [0..1]
direction : SV_ParameterDirection [0..1]
name : MemberName
optionality : CharacterString
repeatability : boolean
valueType : Type
1..*
containsOperations
SV_OperationMetadata
connectPoint : CI_OnlineResource [1..*]
DCP : DCPList [1..*]
invocationName : CharacterString [0..1]
operationDescription : CharacterString [0..1]
operationName : CharacterString
0..*
parameters
name
dependsOn
0..*
Dependencies

Figure 7.10 — Service information model from ISO 19119

7.2.3 WRS capabilities enhancements

The capabilities of a catalogue service that implements the WRS profile are described using the common OWS schemas. However, a profile-specific extension element is defined in order to express some of the specialized features and capabilities introduced in the WRS profile. Listing 7.0 shows the extra information element that augments a basic OGC capabilities document. The relevant schema is listed in Annex E.

Listing 7.0 — WRS capabilities enhancements

```
<csw:Capabilities version=“0.9.0”
  updateSequence=“2004.05.30”
  xmlns:csw=“http://www.opengis.net/cat/csw”
  xmlns:ows=“http://www.opengis.net/ows”>

  <ows:ServiceIdentification>
    <!-- content -->
  </ows:ServiceIdentification>
  <ows:ServiceProvider>
    <!-- content -->
  </ows:ServiceProvider>
  <ows:OperationsMetadata>
    <!-- content -->
  </ows:OperationsMetadata>

  <ExtraServiceInfo xmlns=“http://www.opengis.net/cat/wrs”>
    <feature name="distributed-search" isSupported="true" />
    <feature name="compound-queries" isSupported="false" />
    <property name="capability-class">
      Enhanced Publication
    <property>

    <modules>
      <module
        name="http://www.opengis.net/cat/wrs/modules/standard" />
    </modules>

    <data-formats>
      <data-format contentType="application/xml">
        <schema
          id="http://schemas.opengis.net/wrs/0.9.0/rim"
          schemaLanguage="http://www.w3.org/2001/XMLSchema" />
      </data-format>
      <data-format contentType="text/xml">
        <schema
          id="http://schemas.opengis.net/csw/2.0.0/record"
          schemaLanguage="http://www.w3.org/2001/XMLSchema" />
      </data-format>
      <data-format contentType="image/png" />
    <data-formats>
  </ExtraServiceInfo>

</csw:Capabilities>
```

A standard set of features and properties are used to convey service capabilities that are specific to WRS catalogues; these appear within the service type-specific section of the capabilities document. While feature flags always have boolean values, property values may be simple or complex. All feature and property names are fully-qualified URIs. A few standard ones are defined in this subclause, but vendors may introduce additional ones based on URIs under their control. Table 7.8 defines standard WRS service features; the prefix is “http://www.opengis.net/cat/wrs/features/”.

Table 7.8 — Standard WRS service features

Identifier	**Default**	**Description**
distributed-search	false	Indicates whether the service can perform distributed searches within a federation.
xpath-query	false	Indicates whether the catalogue can query XML resources using XPath expressions .
audit-trail	false	Indicates whether the catalogue maintains an audit trail for transaction requests (see 8.9.3)
xpointer	false	Indicates whether the service supports the W3C XPointer framework and can return fragments of repository items that reflect an XML-based media type.

Table 7.9 defines WRS service properties; the prefix is “http://www.opengis.net/cat/wrs/properties/”.

Table 7.9 — Standard WRS service properties

Identifier	**Description**
capability-class	The capability class supported by the catalogue service (see 7.1)
resultset-timeout	The time interval (in minutes) for which a result set remains active before it expires and is no longer available (see 8.4.3).

7.3 Catalogue information model

7.3.1 Overview of the ebRIM

The WRS information model is based on the ebXML Registry Information Model (ebRIM), version 2.5. The information model is a conceptual model that specifies how (meta)data is organized within the catalogue (i.e. the public schema). The main purpose of the model is to define a formal structure representing catalogued resources and their interrelationships, thereby providing a logical schema for browsing and searching catalogue content. A high-level view of the model—with most of the class attributes suppressed—appears in Figure 7.2. The WRS application profile **does not** make use of any of the behavioural features defined for ebRIM classes—only the attributes are employed as documented in the ebRIM specification.

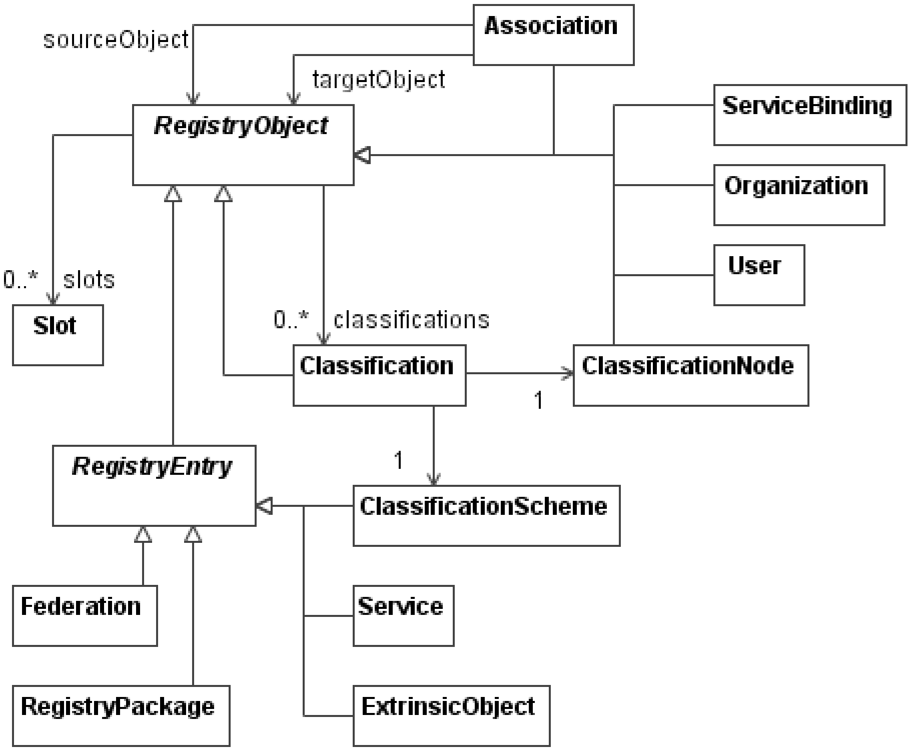

Figure 7.2 — High-level view of ebRIM (details suppressed)

The more detailed UML representations have been organized into the core packages shown in Figure 7.3[1]. The *General* package includes registry objects of general utility; the `RegistryObject` class is the base class for most of the ebRIM. The *Lifecycle* package includes specialized registry objects—registry entries—that add various lifecycle attributes (e.g., stability, version, expiration date). The *Relationships* package includes classes used to classify or link registry objects. The *Events* package focuses on elements needed to maintain an audit trail and to characterize notification events of interest to subscribers.

---

[1] The ebRIM specification does not define any packages—they are introduced here as a convenience to distinguish the purposes served by various model elements.

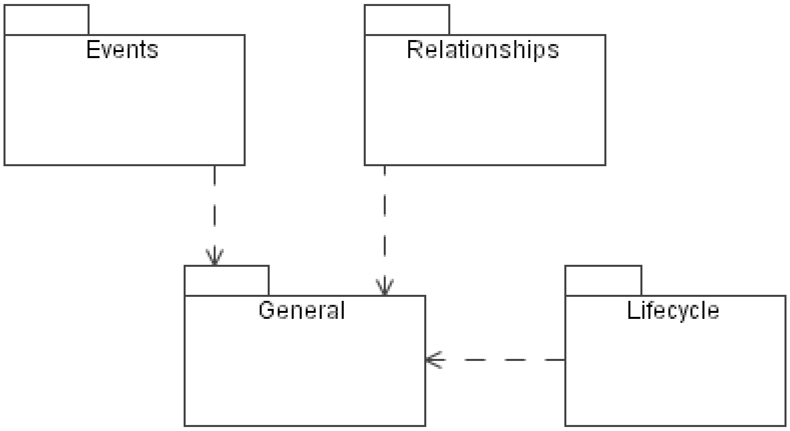

Figure 7.3 — Core ebRIM packages

7.3.2 General package

This package contains the abstract `RegistryObject` base class that defines the attributes common to all registry objects. Table 7.2 briefly describes the classes appearing in Figure 7.4; following each description is a reference to the relevant section of the ebRIM 2.5 specification that provides additional details.

Table 7.2 — Classes in the General package

Class name	Description
RegistryObject	This is the base class for any object held in the Registry which has unique identity. Common attributes such as **`id`**, **`name`** and **`description`** are defined here [ebRIM, 7.5]. The public identifier MUST be an absolute URI; it SHOULD conform to a registered URI scheme[2].
ExternalIdentifier	This defines an identifier for a RegistryObject that is typically derived from external sources (i.e. not assigned by the Registry). Examples are a Social Security Number or an alternative name for a company. The identifier must fall within a ClassificationScheme held by the Registry. [ebRIM, 7.10]
ExternalLink	A Uniform Resource Identifier (URI) used to associate Registry content with an external resource. [ebRIM 7.11]

---

[2] The IANA registry of URI schemes is available at <http://www.iana.org/assignments/uri-schemes>; registered URN namespaces are listed here: <http://www.iana.org/assignments/urn-namespaces>.

Class name	Description
Organization	A representation of an organization associated with the Registry. This includes the organization name and contact information. [ebRIM 7.13]
ServiceBinding	Technical information on how to interact with a particular Service instance. [ebRIM 10.2]
SpecificationLink	A reference to a technical document (i.e. WSDL or CORBA IDL) that describes how to use a particular Service instance. [ebRIM 10.3]
User	A representation of an individual who has registered with the Registry. This includes the user name and contact information. [ebRIM 7.12]

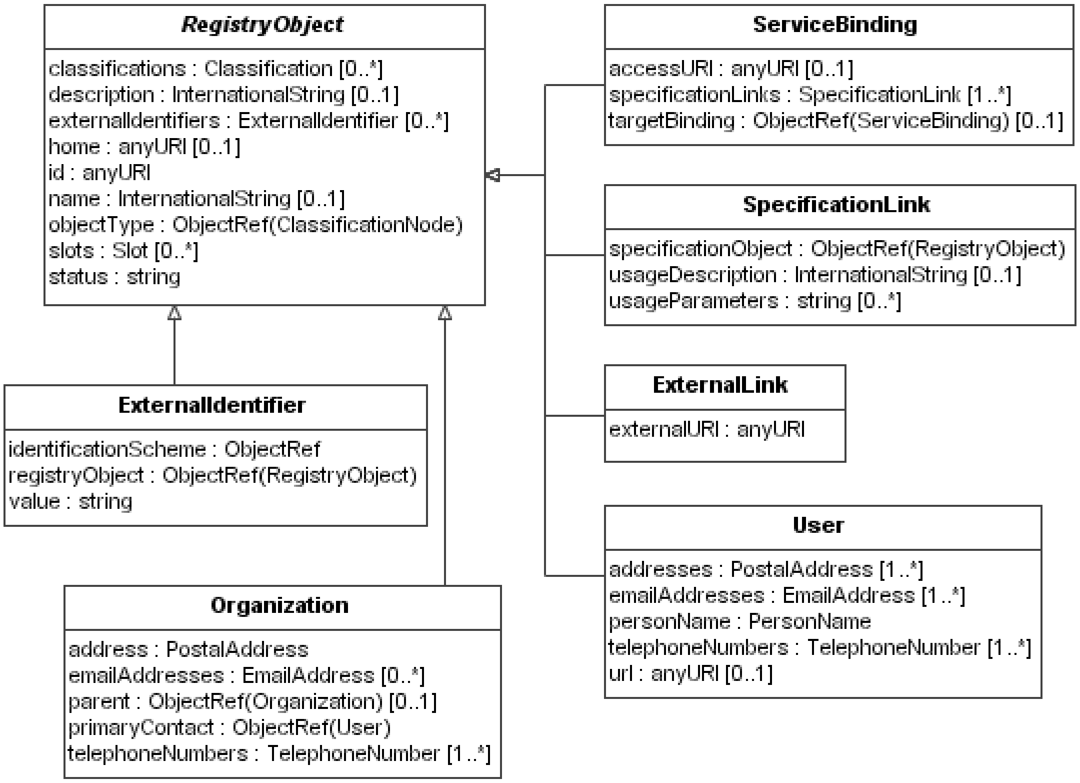
RegistryObject
classifications : Classification [0..*]
description : InternationalString [0..1]
externalIdentifiers : ExternalIdentifier [0..*]
home : anyURI [0..1]
id : anyURI
name : InternationalString [0..1]
objectType : ObjectRef(ClassificationNode)
slots : Slot [0..*]
status : string
ServiceBinding
accessURI : anyURI [0..1]
specificationLinks : SpecificationLink [1..*]
targetBinding : ObjectRef(ServiceBinding) [0..1]
SpecificationLink
specificationObject : ObjectRef(RegistryObject)
usageDescription : InternationalString [0..1]
usageParameters : string [0..*]
ExternalLink
externalURI : anyURI
ExternalIdentifier
identificationScheme : ObjectRef
registryObject : ObjectRef(RegistryObject)
value : string
User
addresses : PostalAddress [1..*]
emailAddresses : EmailAddress [1..*]
personName : PersonName
telephoneNumbers : TelephoneNumber [1..*]
url : anyURI [0..1]
Organization
address : PostalAddress
emailAddresses : EmailAddress [0..*]
parent : ObjectRef(Organization) [0..1]
primaryContact : ObjectRef(User)
telephoneNumbers : TelephoneNumber [1..*]

Figure 7.4 — Elements of the General package

7.3.3 Lifecycle package

This package contains the abstract `RegistryEntry` base class that defines the attributes common to all registry objects subject to lifecycle management. Table 7.3

briefly describes the classes appearing in Figure 7.5; following each description is a reference to the relevant section of the ebRIM 2.5 specification that provides additional details.

Table 7.3 — Classes in the Lifecycle package

Class name	Description
RegistryEntry	A base class for lifecycle management of high level Registry objects. Attributes include expiration date and version numbering. [ebRIM 7.6]
ClassificationScheme	A registered taxonomy used to classify RegistryObjects. A ClassificationScheme may be defined entirely within the Registry or it may be defined externally. [ebRIM 9.1]
ExtrinsicObject	Used to assign type information (i.e. mimeType) to a RegistryEntry when its type is not known to the Registry. [ebRIM 7.8]
Federation	Acts as a grouping mechanism for cooperating Registry instances. [ebRIM 12.2]
Registry	This class represents a cooperating Registry. Registries may cooperate for a variety of reasons, including: distributed searching, notification, replication etc. [ebRIM 12.1]
RegistryPackage	This class provides a means to logically group RegistryObjects. [ebRIM 7.9]
Service	A flexible way to define a network accessible service offering. A variety of service types can be described, including: web services (WSDL), CORBA and Z39.50. [ebRIM 10.1]

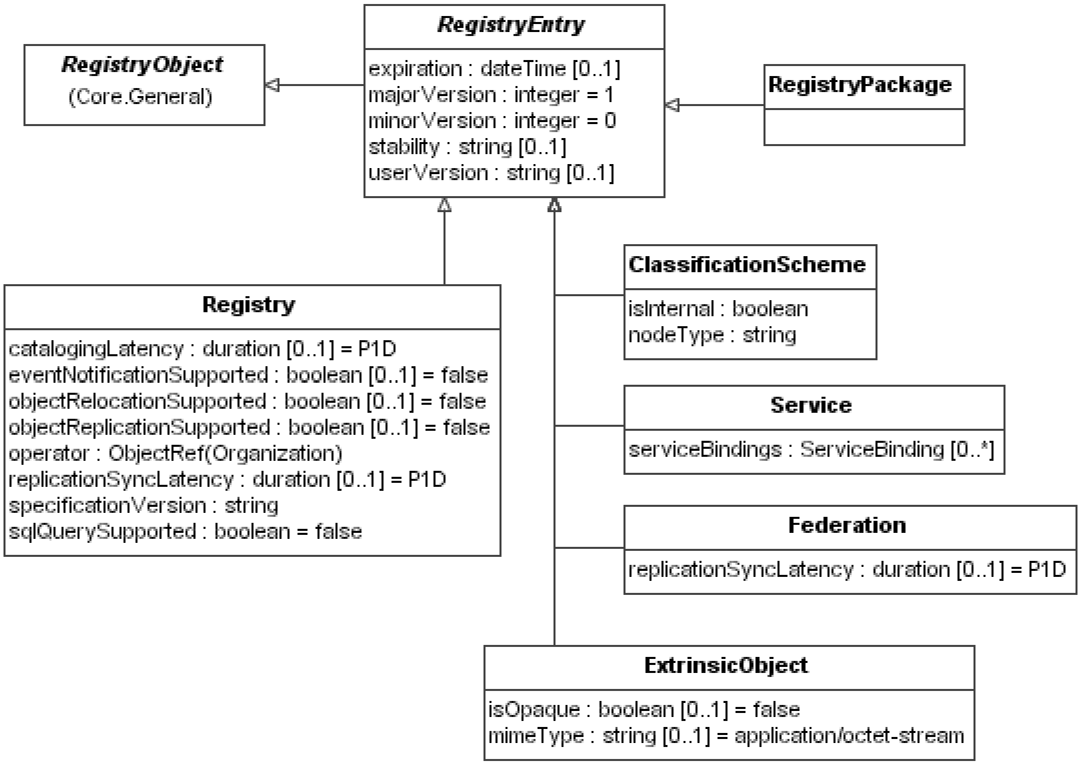
RegistryObject
(Core.General)
RegistryEntry
expiration : dateTime [0..1]
majorVersion : integer = 1
minorVersion : integer = 0
stability : string [0..1]
userVersion : string [0..1]
RegistryPackage
Registry
catalogingLatency : duration [0..1] = P1D
eventNotificationSupported : boolean [0..1] = false
objectRelocationSupported : boolean [0..1] = false
objectReplicationSupported : boolean [0..1] = false
operator : ObjectRef(Organization)
replicationSyncLatency : duration [0..1] = P1D
specificationVersion : string
sqlQuerySupported : boolean = false
ClassificationScheme
isInternal : boolean
nodeType : string
Service
serviceBindings : ServiceBinding [0..*]
Federation
replicationSyncLatency : duration [0..1] = P1D
ExtrinsicObject
isOpaque : boolean [0..1] = false
mimeType : string [0..1] = application/octet-stream

Figure 7.5 — Elements of the Lifecycle package

7.3.4 Relationships package

This package contains the abstract `RegistryObject` base class that defines the attributes common to all registry objects. Table 7.4 briefly describes the classes appearing in Figure 7.6; following each description is a reference to the relevant section of the ebRIM 2.5 specification that provides additional details.

Table 7.4 — Classes in the Relationships package

Class name	Description
Association	This class describes a directed relationship between two RegistryObjects. Associations are flexible and can be used to describe many-to-many relationships between objects within the Registry. [ebRIM 8.9]
Classification	A link between a RegistryObject and a particular category (or ClassificationNode) within a ClassificationScheme. [ebRIM 9.3]
ClassificationNode	A category within a taxonomy (or ClassificationScheme). [ebRIM 9.2]

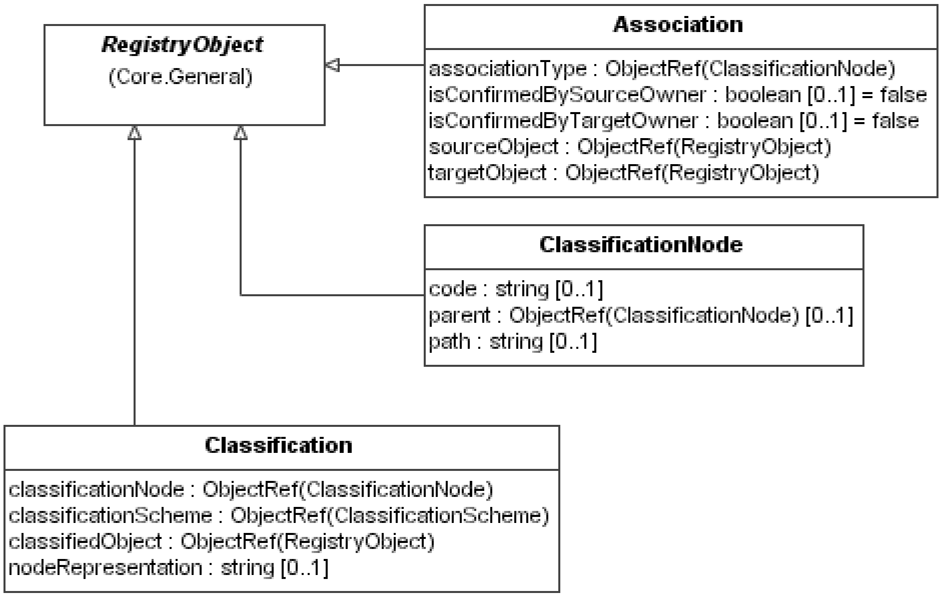
RegistryObject
(Core.General)
Association
associationType : ObjectRef(ClassificationNode)
isConfirmedBySourceOwner : boolean [0..1] = false
isConfirmedByTargetOwner : boolean [0..1] = false
sourceObject : ObjectRef(RegistryObject)
targetObject : ObjectRef(RegistryObject)
ClassificationNode
code : string [0..1]
parent : ObjectRef(ClassificationNode) [0..1]
path : string [0..1]
Classification
classificationNode : ObjectRef(ClassificationNode)
classificationScheme : ObjectRef(ClassificationScheme)
classifiedObject : ObjectRef(RegistryObject)
nodeRepresentation : string [0..1]

Figure 7.6 — Elements of the Relationships package

7.3.5 Events package

This package contains the abstract `RegistryObject` base class that defines the attributes common to all registry objects. Table 7.5 briefly describes the classes appearing in Figure 7.7; following each description is a reference to the relevant section of the ebRIM 2.5 specification that provides additional details.

Table 7.5 — Classes in the Events package

Class name	Description
Action	A base class for event triggered actions within the Registry. [ebRIM 11.4]
AdhocQuery	A mechanism for storing queries which is similar in purpose to stored procedures in a relational database system. A common use is to select events of interest to users for use in their subscriptions. [ebRIM 11.3]
AuditableEvent	Provides a record of actions that modify content or ownership of RegistryObjects. [ebRIM 11.1]
NotifyAction	The action to notify a user whenever their subscription request has been triggered. [ebRIM 11.5]

Class name	Description
Subscription	Mechanism whereby users can request notification of specific events. These are interval based and use AdHocQuery for event selection. [ebRIM 11.2]

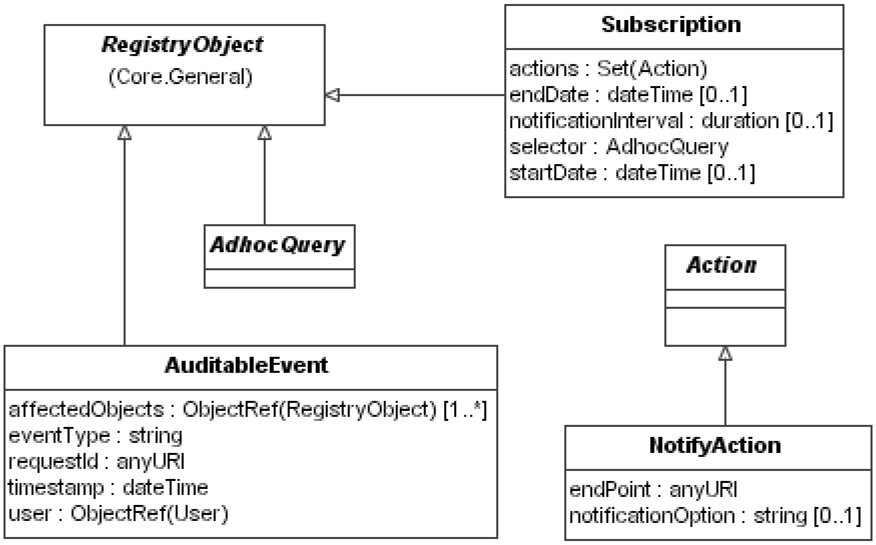

Figure 7.7 — Elements of the Events package

7.3.6 WRS extensions to ebRIM

The WRS package introduces a few extension elements in order to meet common requirements in the geospatial domain. These extensions constitute a thin “customization layer” as shown in Figure 7.8 using UML notation. The corresponding XML schema appears in Annex E.

The `WRSExtrinsicObject` class adds the *content* attribute in order to specify the location of some representation of a resource available in a repository that may not be managed by the catalogue service. The getExtrinsicContent() operation returns the content of the resource as the entity body within an HTTP response message. The Content-Type header MUST have the same value as the ExtrinsicObject.mimeType attribute. The ETag header MAY convey the value of the RegistryObject.id attribute.

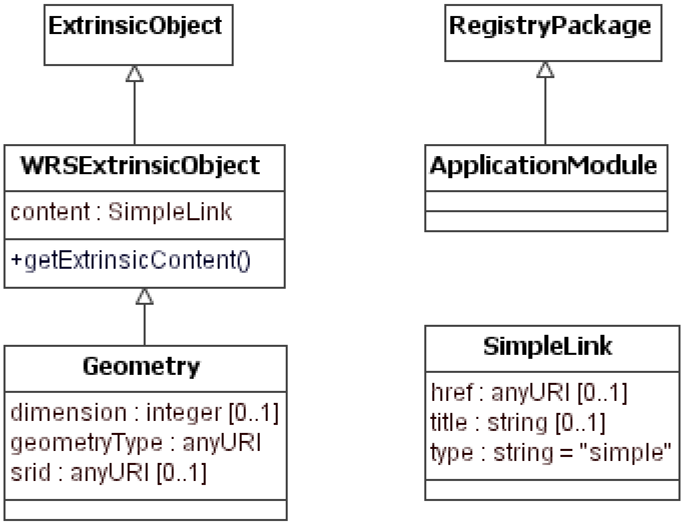

Figure 7.8 — Elements of the WRS extensions package

The `Geometry` class may be used to provide the value of some geometric property of a registry object. It extends `WRSExtrinsicObject` and adds a few attributes based on the simple geometry model defined in OGC 99-049 ("OpenGIS Simple Features Specification for SQL"); these are defined in Table 7.6. The inherited getExtrinsicContent() operation returns a GML 3.0 representation of the geometry instance (OGC 02-023r4).

Table 7.6 — Geometry attributes

Attribute name	Definition
dimension	The inherent dimension of the Geometry instance, which must be less than or equal to the coordinate dimension. The value must be an integer in the range [0-2].
geometryType	Identifies the concrete primitive or aggregate subtype of the geometry instance. The value must be a URI referencing a relevant type definition.
srid	Identifies the spatial reference system for the geometry instance. The value must be a URI referencing a definition of the reference system.

The `ApplicationModule` class provides a simple means of defining a set of domain- or discipline-specific extensions that describe additional kinds of content handled by the catalogue service; it extends `RegistryPackage` but currently adds no other attributes. However, it *does* include the getMemberObjects() operation inherited from the ebRIM

`RegistryPackage` class. See subclause 7.2.7 for additional information about using modules.

7.3.7 Application modules

The ebRIM offers a number of extensibility points that permit it to be tailored for specific purposes. The extensibility points include:

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

The term “application module” refers to the specification of the syntax and semantics of a cohesive set of registry objects that pertain to a specific application domain—it basically constitutes a catalogue expansion pack. For example, a ‘Web Mapping’ module might include elements for working with the style descriptors and symbol collections used in map production. A ‘Geodesy’ module will include elements for cataloguing coordinate reference systems and their components.

Since a module is a type of `RegistryPackage`, it inherits all the attributes of a registry entry. The XML syntax for defining a module is shown schematically in Listing 7.1. Such a complete representation would be returned within a “full” response, but some details are omitted for partial element sets (i.e. “brief” or “summary”). Module members include the following types of registry objects: ClassificationScheme, ClassificationNode, Association, StoredQuery, or Document; these allowable member types are summarized in Table 7.7. Any number of these registry objects may be included in any order within the RegistryObjectList element.

Listing 7.1 — Full XML representation of an ApplicationModule

```
<wrs:ApplicationModule
  id=“some-uri”
  objectType=“urn:opengis:spec:catalogue.csw:object-
      types:CSWModule”
  majorVersion=“positiveInteger”
  minorVersion=“positiveInteger”
  userVersion=“1.0rc2”
  xmlns:wrs=“http://www.opengis.net/cat/wrs”
  xmlns:rim=“urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5”>

  <rim:Name><!-- localized strings --></rim:Name>
  <rim:Description><!-- localized strings --></rim:Description>
  <rim:Slot name=“name” slotType=“parentTypeRef”>
    <rim:ValueList><!-- value domain --><rim:ValueList>
  </rim:Slot>
  <rim:RegistryObjectList>
    <!-- list of member registry objects -->
  </rim:RegistryObjectList>

</wrs:ApplicationModule>
```

Table 7.7 — Registry object members of an application module

Object type	Purpose
ClassificationScheme	Defines a new classification scheme (e.g. taxonomy, thesaurus) to characterize catalogue content
ClassificationNode	Extends a classification scheme or taxonomy
Association	Introduces new kinds of relationships; this is a *meta-association* that links the types of objects (i.e., classification nodes) that may participate in the relationship.
StoredQuery	A named query that embodies a common search pattern as a convenience for clients.
Document	User documentation for the module
Stylesheet	A presentation resource that may be used to style another resource or transform it into an alternative format.

There is one predefined application module: the ‘Standard’ module is built into all WRS implementations; it defines resources of general utility, together with some basic constraints on their usage (Annex F). The Standard module includes:

- a taxonomy of standard WRS object types that includes the standard ebRIM types;
- the ISO 19119 service classification scheme;
- associations that interrelate instances of the WRS object types;
- User documentation in simplified DocBook format; this document may also be requested in an alternative format (e.g. application/xhtml+xml for display in a web browser);
- definitions of commonly used slots—these are based on standard Dublin Core metadata terms[3].

7.4 Supported data formats

7.4.1 Common CSW record format

Clause 10 of the OpenGIS Catalogue Services 2.0 specification defines a common record format for all web-based catalogue services. A csw:Record instance is an XML document that contains a collection of Dublin Core metadata terms. The property set that constitutes

---

[3] The latest specification of DCMI terms is available at <http://dublincore.org/documents/dcmi-terms/>.

a “summary” view is shown in Table 7.6. The “brief” view includes only the *identifier* and *type* elements. The “full” view may include any of the terms, plus the ows:BoundingBox element from OWS Common Implementation Specification (OGC-04-016).

Table 7.6 — Common CSW properties: Summary property set (Dublin Core terms)

Common CSW property	Definition
identifier	An unambiguous reference to the resource within a given context.
type	The nature or genre of the content of the resource.
title	A name given to the resource.
subject	The topic of the content of the resource.
format	The physical or digital manifestation of the resource.
relation	A reference to a related resource. A relation element has two components which are separated by a semi-colon: *`role-name`*; *`target-URI`* , where `role-name` identifies the nature of the relationship and `target-URI` identifies the associated resource.
modified	Date on which the resource was changed.
abstract	A summary of the content of the resource.
spatial	Spatial characteristics of the intellectual content of the resource.

Table 7.7 indicates how selected CSW properties are mapped to properties of the WRS information model. If no mapping is explicitly specified, the property maps to a Slot of the same name.

Table 7.7 — Mapping CSW properties to the WRS information model

Common CSW property	WRS property
identifier	RegistryObject/@id
type	RegistryObject/@objectType
title	RegistryObject/Name
format	ExtrinsicObject/@mimeType
BoundingBox	Geometry

Common CSW property	WRS property
relation	Association/@targetObject NOTE: consider mapping the 'role' part in a DCSV encoding to Association/@associationType, where `<relation>uri-1; role=operatesOn</relation>`
valid	RegistryEntry.expiration
TBD	

7.4.2 Default representation: ebRIM 2.5

The default format for representing WRS registry objects is XML (application/xml) that is valid with respect to the WRS customization schema (Annex C); this schema imports the ebRIM v2.5 schema (Annex B). Figure 7.9 is a UML package diagram illustrating the dependencies among the CSW and WRS schemas. The WRS schemas are briefly described in Table 7.8.

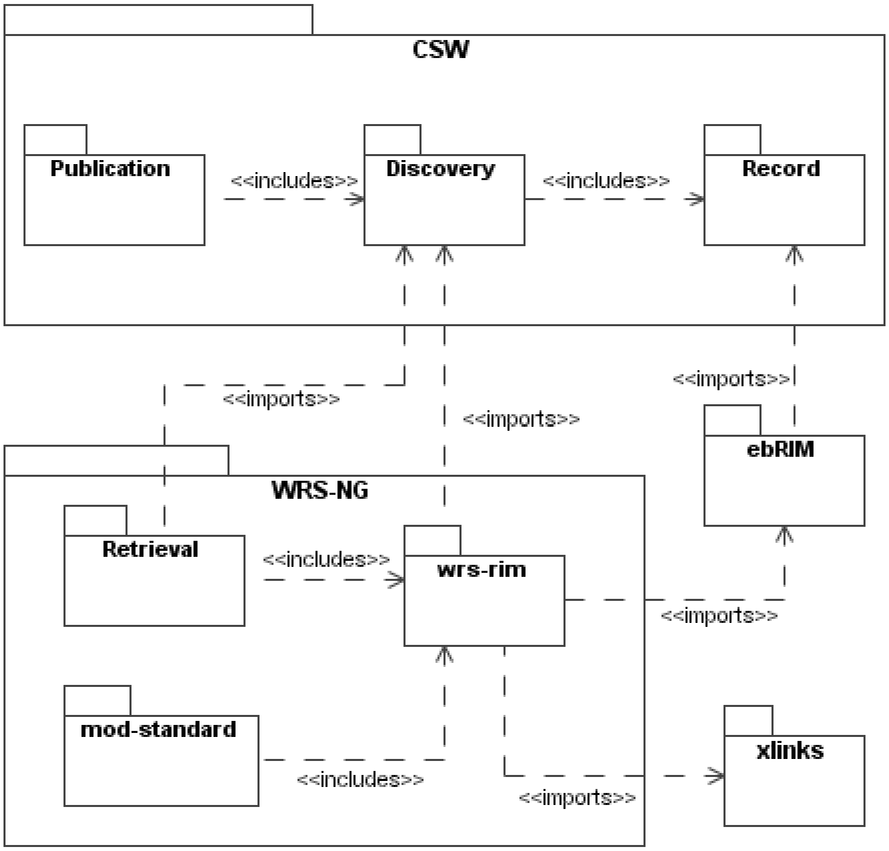

Figure 7.9 — Schema dependencies

Table 7.8 — The WRS schema set

Schema property	Value
identifier	http://www.oasis-open.org/committees/regrep/documents/2.5/schema/rim.xsd
target namespace	urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5
description	Defines the XML representation of ebRIM 2.5 objects (see Annex B).
identifier	http://schemas.opengis.net/wrs/0.9.0/rim
target namespace	http://www.opengis.net/cat/wrs
description	Defines WRS extensions to ebRIM (see Annex C).
identifier	http://schemas.opengis.net/wrs/0.9.0/retrieval
target namespace	http://www.opengis.net/cat/wrs
description	Defines additional WRS message elements (see Annex D).
identifier	http://schemas.opengis.net/wrs/0.9.0/modules/standard
target namespace	http://www.opengis.net/cat/wrs
description	Declares elements that instantiate new extrinsic object types for the Standard WRS module (see Annex F).
identifier	http://schemas.opengis.net/wrs/0.9.0/wrs-extras
target namespace	http://www.opengis.net/cat/wrs
description	Declares elements that extend various common OWS schema components (see Annex E).

7.5 Native language support

Some localization is accommodated through the use of the `InternationalString` class that is the value of the RegistryObject.name and RegistryObject.description attributes. An instance of this class is composed of a collection of `LocalizedString` instances that convey language-specific values for these attributes. Listing 7.2 provides a simple example of multilingual descriptors.

Listing 7.2 — Support for multilingual descriptors

```
<wrs:WRSExtrinsicObject
  id="some-uri"
  objectType="urn:opengis:spec:catalogue.csw:object-
      types:Dataset"
  majorVersion="positiveInteger"
  minorVersion="positiveInteger"
  userVersion="200206"
  xmlns:wrs=“http://www.opengis.net/cat/wrs”
  xmlns:rim=“urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5”
  xmlns:xml=“http://www.w3.org/XML/1998/namespace”>

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

</wrs:WRSExtrinsicObject>
```

8 External interfaces

8.1 Overview

A number of external interfaces may be realized by a WRS implementation, as shown in Figure 7.10. Some these interfaces are inherited from the *OpenGIS Catalogue Services 2.0 Specification* (OGC-04-021) and the *OWS Common Implementation Specification* (OGC-04-016). Consult these documents for details.

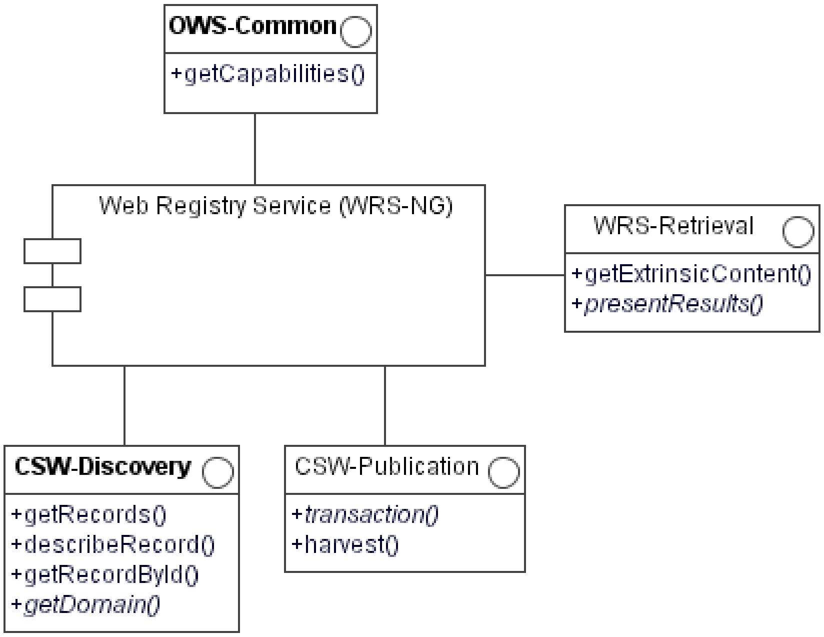

Figure 7.10 — WRS interfaces

The interfaces named in **bold** are mandatory—they must be supported by a conforming implementation. In some cases not all operations defined for an interface need be implemented; abstract operations—named in *italics*—are deemed optional. If a service does not implement an abstract operation, it must respond with a service exception and return an `UnsupportedOperation` code.

8.2 HTTP method bindings

The HTTP/1.1 specification (RFC 2616) defines eight methods for manipulating and retrieving representations of resources. Only the GET and POST methods are required to implement this application profile. All of the operations defined in this application profile MUST be bound to HTTP methods as summarized in Table 8.3, but alternative bindings may also be specified for a specific service instance. For example, an operation bound to the GET method may also be bound to the POST method.

Table 8.3 — Mandatory HTTP method bindings

WRS operation	**HTTP method binding**
OWS-Common.getCapabilities()	GET
CSW-Discovery.getRecords()	POST
CSW-Discovery.describeRecord()	POST
CSW-Discovery.getDomain()	GET
CSW-Discovery.getRecordById()	GET
CSW-Publication.transaction()	POST
CSW-Publication.harvest()	POST
WRS-Retrieval.presentResults()	POST
WRS-Retrieval.getExtrinsicContent()	GET

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

8.3 OWS-Common interface

8.3.1 Operations provided

The `OWS-Common` interface provides operations that are common to all OGC web services. Currently only one operation is defined: getCapabilities(). This interface is mandatory for all conforming implementations. The UML diagram in Figure 8.2.1 summarizes the interface signature.

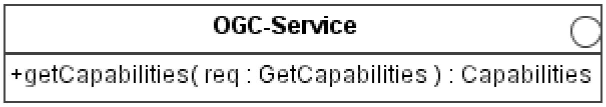

Figure 8.2.1 — OGC-Service interface signature

8.3.2 getCapabilities()

8.3.2.1 Purpose

The getCapabilities operation is used to retrieve a description of service capabilities; the service description includes both the operational and non-operational characteristics of the service.

See the *OWS Common Implementation Specification* (OGC-04-016) for details

8.3.2.2 Exceptions

No fault messages are associated with a getCapabilities request. If a section is specified and it is incorrectly identified, the entire capabilities document for the service MUST be returned.

8.3.2.3 Usage guide

The getCapabilities operation must be bound to at least the HTTP GET method. The section element is serialized as a name-value pair within the query component of the request URI as described in the HTTP/1.1 standard [RFC 2616].

8.4 CSW-Discovery interface

8.4.1 Operations provided

The `CSW-Discovery` interface provides various operations for searching and retrieving catalogue content in a synchronous or asynchronous manner. Search and retrieval may be performed as separate actions, or matching catalogue entries can be included in the response message if desired. This interface is mandatory for all conforming implementations. The UML diagram in Figure 8.3.1 summarizes the interface signature.

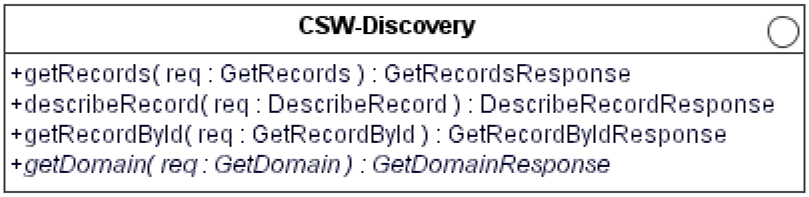
CSW-Discovery
+getRecords(req : GetRecords) : GetRecordsResponse
+describeRecord(req : DescribeRecord) : DescribeRecordResponse
+getRecordById(req : GetRecordById) : GetRecordByIdResponse
+getDomain(req : GetDomain) : GetDomainResponse

Figure 8.3.1 — CSW-Discovery interface signature

The constituent operations are described in clause 10 of the *OpenGIS Catalogue Services 2.0 Specification* (OGC-04-021). Any extensions or restrictions introduced by the WRS profile are documented in the following subclauses.

8.4.2 getRecords()

8.4.2.1 Purpose

The getRecords operation is the principal means of searching catalogue content. The request specifies either a single query, a stored query, or a compound query (i.e. a set operator). The client may assign a request identifier (an absolute URI) for tracking purposes. A distributed search is performed if the <DistributedSearch> element is included within the request.

The client may request that some or all of the registry objects in the result set be "piggybacked" in the response message. The various retrieval control attributes govern this behaviour. If the maxRecords attribute has the value "0", then only a summary of the result set is included in the response; the client can subsequently retrieve members of the result set by submitting a presentResults request.

8.4.2.2 Query extensions

A GetRecords request must contain a query statement. The OGC filter syntax [OGC 02-059] must be understood by all WRS catalogue services—this is the default syntax for specifying an ad hoc query statement. In addition to the csw:Query statement documented in the Catalogue 2.0 specification, a WRS catalogue MAY also process XPath query expressions; such a query can substitute for the csw:AbstractQuery element in a GetRecords request. The XPath expression is evaluated against the content of extrinsic objects that have an XML-based media type and for which the *isOpaque* property has the value "false". The result set is populated with those objects for which there is a non-empty XPath result. Listing 8.100 includes the relevant type definition from the schema in Annex B.

Listing 8.100 — XPathQuery type definition

```
<xsd:complexType name="XPathQueryType" id="XPathQueryType">
  <xsd:complexContent>
    <xsd:extension base="csw:AbstractQueryType">
      <xsd:sequence>
        <xsd:element name="elementSet" type="csw:ElementSetType"
          minOccurs="0" default="brief" />
         <xsd:element name="context" type="xsd:string" />
         <xsd:element name="xpath" type="xsd:string" />
      </xsd:sequence>
      <xsd:attribute name="version" type="xsd:string" use="required" />
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The message information items are described in Table 8.105. Information items named in **bold** are required.

Table 8.105 — XPathQuery: information items

Name	Type	Description
version	attribute	version of XPath grammar
elementSet	element	The element set to return (a “summary” view by default)
context	element	The context node for the query
xpath	element	The expression to be evaluated

8.4.2.3 Binding variables

Binding variables—or aliases—may be declared in a csw:Query statement so as to avoid ambiguity when specifying complex queries that navigate associations by traversing links between related registry objects. The value of the Query/@typeNames attribute is a whitespace-separated list of object types to query. The basic syntax is

`objectType=var1,var2,...,varN`

where each variable ranges over instances of type `objectType`. Multiple variables may be bound to a given object type; these are delimited by a comma as shown above. The variables are subsequently referenced as $varN. Listing 8.101 includes an example to search for dataset descriptions that are 1) associated with a particular service and that 2) conform to a specified application schema.

Listing 8.101 — Using a binding variable in a query statement

```
<?xml version='1.0' encoding='UTF-8'?>
<GetRecords version='2.0.0' xmlns='http://www.opengis.net/cat/csw'
  xmlns:ogc='http://www.opengis.net/ogc'
  startPosition='1'
  maxRecords='10'
  outputFormat='application/xml; charset=UTF-8'>

  <Query typeNames='Service Association=a1,a2 Dataset'>
    <ElementSetName typeNames='Dataset'>full</ElementSetName>
    <Constraint version='1.0.0'>
      <ogc:Filter>
        <ogc:And>
          <ogc:PropertyIsEqualTo>
            <ogc:PropertyName>$a1/@associationType</ogc:PropertyName>
            <ogc:Literal>operatesOn</ogc:Literal>
          </ogc:PropertyIsEqualTo>
          <ogc:PropertyIsEqualTo>
            <ogc:PropertyName>$a1/@sourceObject</ogc:PropertyName>
            <ogc:Literal>
              urn:uuid:86084c6a-292f-4687-bf52-aef49b5ff2d6
            </ogc:Literal>
          </ogc:PropertyIsEqualTo>
          <ogc:PropertyIsEqualTo>
            <ogc:PropertyName>Dataset/@id</ogc:PropertyName>
            <ogc:PropertyName>$a1/@targetObject</ogc:PropertyName>
          </ogc:PropertyIsEqualTo>
          <ogc:PropertyIsEqualTo>
            <ogc:PropertyName>Dataset/@id</ogc:PropertyName>
            <ogc:PropertyName>$a2/@sourceObject</ogc:PropertyName>
          </ogc:PropertyIsEqualTo>
          <ogc:PropertyIsEqualTo>
            <ogc:PropertyName>$a2/@associationType</ogc:PropertyName>
            <ogc:Literal>applicationSchemaInfo</ogc:Literal>
          </ogc:PropertyIsEqualTo>
          <ogc:PropertyIsEqualTo>
            <ogc:PropertyName>$a2/@targetObject</ogc:PropertyName>
            <ogc:Literal>
              urn:uuid:01910ba2-66af-44b6-8486-8693ece6633c
            </ogc:Literal>
          </ogc:PropertyIsEqualTo>
        </ogc:And>
      </ogc:Filter>
    </Constraint>
  </Query>
</GetRecords>
```

If no exceptional conditions arise, the service returns a GetRecordsResponse message that conforms to the schema in Listing 8.103. While status information is always included, search results are returned only if the request specified maxRecords>0.

8.4.2.4 Exceptions

An exception message is returned if an abnormal condition arises while processing a getRecords request message. These are encapsulated in a ServiceExceptionReport; the fault codes and conditions are listed in the table below.

Table 8.106 — Exceptions associated with GetRecords

Fault code	Fault subcode	Condition
ows:Client	wrs:InvalidRequest	The request message is either invalid or is not well-formed.

8.4.3 getRecordById()

8.4.3.1 Overview

The getRecordById() operation may be invoked to request the default representation of a specified registry object. If there is a matching registry object, the default representation is returned. If there is no matching record, the root element of the response message is empty (i.e., there is no child record).

This operation is provided as a convenience to clients, as it corresponds to a getRecords request where the filter expression specifies a registry object identifier; however, getRecordById is typically bound to the HTTP GET method.

See subclause 10.9 of *OpenGIS Catalogue Services 2.0 Specification* (OGC-04-021) for details.

8.4.3.2 Exceptions

An exception is returned if an abnormal condition arises while processing a request message. These are encapsulated in a ServiceExceptionReport; the fault codes and conditions are listed in the table below.

Table 8.500 — Exceptions associated with getRecordById

Fault code	Fault subcode	Condition
ows:Client	wrs:InvalidRequest	The request message is either invalid or is not well-formed.

8.4.3.3 Usage guide

The getRecordById operation must be bound to at least the HTTP GET method. The id and elementSet elements are serialized as name-value pairs within the query component of the request URI as described in the HTTP/1.1 standard [RFC 2616].

8.4.4 describeRecord()

8.4.4.1 Overview

The describeRecord operation provides a simple means of searching and retrieving formal descriptions of the information model supported by the catalogue, including extensions.

See subclause 10.6 of *OpenGIS Catalogue Services 2.0 Specification* (OGC-04-021) for details.

8.4.4.2 Exceptions

A exception is returned if an abnormal condition arises while processing a request message. These are encapsulated in a ServiceExceptionReport; the fault codes and conditions are listed in the table below.

Table 8.600 — Exceptions associated with describeRecord

Fault code	Fault subcode	Condition
ows:Client	wrs:InvalidRequest	The request message is either invalid or is not well-formed.

8.4.5 getDomain()

8.4.5.1 Overview

The getDomain operation describes the value domain of a specified data element or message parameter, where the value domain is the set of actual or *permissible* values. The value domain may be enumerated or non-enumerated[4].

The operation is optional. See subclause 10.7 of *OpenGIS Catalogue Services 2.0 Specification* (OGC-04-021) for details.

8.4.5.2 Exceptions

An exception is returned if an abnormal condition arises while processing a request message. These are encapsulated in a ServiceExceptionReport; the fault codes and conditions are listed in the table below.

Table 8.100 — Exceptions associated with getDomain

Fault code	Fault subcode	Condition
ows:Client	wrs:InvalidRequest	The request message is either invalid or is not well-formed.
ows:Server	wrs:UnsupportedOperation	The catalogue does not support the operation.

---

[4] For example, ISO 3166 describes a set of seven enumerated value domains for the conceptual domain "Countries of the World": short name in English, official name in English, alpha-2 code, alpha-3 code, numeric code, etc.

8.5 WRS-Retrieval interface

8.5.1 Operations provided

The `WRS-Retrieval` interface extends the `CSW-Discovery` interface to provide more specialized operations for searching and retrieving catalogue content. For example, a client may request the resource associated with an extrinsic object. In addition, the catalogue may maintain the results of a query for subsequent retrieval requests; however, unless the service provides some sort of session management facility, a result set only remains available for a limited period as determined by the timeout interval.

This presentResults() operation is optional—a conforming implementation is not required to manage active result sets; such a capability is most useful in the context of performing distributed searches that might take some time to complete. The UML diagram in Figure 8.4.1 summarizes the interface signature. Each operation is described in the following subclauses.

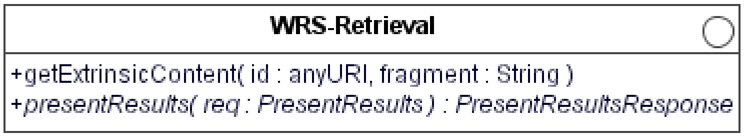
WRS-Retrieval
+getExtrinsicContent(id : anyURI, fragment : String)
+presentResults(req : PresentResults) : PresentResultsResponse

Figure 8.4.1 — WRS-Retrieval interface signature

8.5.2 presentResults()

8.5.2.1 Overview

The presentResults operation allows a client to retrieve the members of a given result set in a particular presentation format. A variety of basic retrieval options may be specified by the client. For example, a slice of the result set may be returned thereby allowing the client to "page through" large result sets. Simple sorting is also possible.

Every result set has a timeout interval—this is a system property determined by the service provider; when this interval has elapsed the result set expires and is no longer available. A catalogue provider may elect to support some kind of session management, in which case the lifetime of the result set is tied to the session context.

8.5.2.2 Operation signature

The PresentResults request message is defined by the XML Schema fragment in Listing 8.401; this is a component of the complete schema listed in Annex D. The message information items are described in Table 8.401, where the attribute group reference has been resolved to include the component attributes that specify retrieval options.

Listing 8.401 — PresentResults type definition

```
<xsd:complexType name="PresentResultsType"
  id="PresentResultsType">
```

```
    <xsd:sequence>
      <xsd:element name="resultSetId" type="xsd:anyURI" />
      <xsd:element ref="ogc:SortBy" minOccurs="0" />
    </xsd:sequence>
    <xsd:attribute name="elementSet" type="csw:ElementSetType"
      use="optional" />
    <xsd:attributeGroup ref="csw:BasicRetrievalOptions" />
</xsd:complexType>
```

Table 8.401 — PresentResults request message: information items

Name	Type	Description
outputFormat	attribute	preferred media type of the response (default value: “application/xml; charset=UTF-8”)
outputSchema	attribute	preferred schema for the retrieval records
startPosition	attribute	starting record in slice (default value: 1)
maxRecords	attribute	the maximum number of records to return (default value: 10)
elementSet	attribute	The element set to return (default value: “summary”)
resultSetId	element	Absolute URI identifying the result set
SortBy	element	A sorting criterion

The three pre-defined element sets are “brief”, “summary” and “full”. The properties bundled within each set are indicated below:

- *brief* – includes the following registry object properties: RegistryObject.id, Registry.objectType, RegistryObject.status (this effectively functions as an object reference)
- *summary* – includes the “brief” properties plus: RegistryObject/Description (preferred lang), RegistryObject/Name (preferred lang), RegistryObject/Slot
- *full* – includes all information items defined for the registry object type

If no error conditions arise, the catalogue returns a PresentResultsResponse message that conforms to the csw:SearchResultsType defined in the Catalogue Services 2.0 specification (reproduced as Listing 8.402 for convenience).

Listing 8.402 — SearchResultsType from the OGC Catalogue 2.0 specification

```
<xsd:complexType name="SearchResultsType" id="SearchResultsType">
  <xsd:sequence>
    <xsd:element ref="csw:AbstractRecord" minOccurs="0"
      maxOccurs="unbounded" />
  </xsd:sequence>
  <xsd:attribute name="resultSetId" type="xsd:anyURI" use="optional" />
  <xsd:attribute name="numberOfRecordsMatched"
    type="xsd:nonNegativeInteger" use="required"/>
  <xsd:attribute name="numberOfRecordsReturned"
```

```
    type="xsd:nonNegativeInteger" use="required" />
  <xsd:attribute name="elementSet" type="csw:ElementSetType"
    use="optional"/>
  <xsd:attribute name="recordSchema" type="xsd:anyURI" use="optional"/>
  <xsd:attribute name="nextRecord" type="xsd:nonNegativeInteger"
      use="optional" />
  <xsd:attribute name="expires" type="xsd:dateTime" use="optional" />
</xsd:complexType>
```

8.5.2.3 Exceptions

A exception is returned if an abnormal condition arises while processing a PresentResults request message. These are encapsulated in a ServiceExceptionReport; the fault codes and conditions are listed in the table below.

Table 8.402 — Exceptions associated with PresentResults

Fault code	Fault subcode	Condition
ows:Client	wrs:InvalidRequest	The request message is either invalid or is not well-formed.
ows:Client	wrs:ResultSetNotFound	The specified result set identifier does not correspond to an active result set.
ows:Server	wrs:UnsupportedOperation	The catalogue does not support the operation.

8.5.2.4 Usage guide

The presentResults operation must be bound to at least the HTTP POST method. The body of the request message is a <PresentResults> element structured in accord with the type definition in Listing 8.401. The body of the response message is a <PresentResultsResponse> element structured in accord with the type definition in Listing 8.402.

8.5.3 getExtrinsicContent()

8.5.3.1 Overview

The getExtrinsicContent operation provides a means of retrieving the content of the resource corresponding to a specified extrinsic object. The content type of the resource is indicated by the value of the WRSExtrinsicObject.mimeType property; in general this is an IANA-registered Internet media type: an XML resource, a PDF document, a JPEG image, an MPEG video clip, etc. A catalogue may recognize only a subset of these media types, as specified in the service capabilities document.

8.5.3.2 Operation signature

The getExtrinsicContent request message is defined by the XML Schema fragment in Listing 8.601. The message information items are described in Table 8.601.

Listing 8.601 — GetExtrinsicContent message type definition

```
<xsd:complexType name="GetExtrinsicContentType"
  id="GetExtrinsicContentType">
  <xsd:sequence>
    <xsd:element name="id" type="xsd:anyURI" />
    <xsd:element name="fragment" type="xsd:string" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

Table 8.601 — GetExtrinsicContent request message: information items

Name	Type	Description
id	element	The identifier for the parent extrinsic object.
fragment	element	A fragment identifier, the syntax and semantics of which is dependent upon the media type of the resource (e.g., the XPointer syntax for XML resources).

If no error conditions arise, the resource is returned as the entity-body in the HTTP response message. In some cases the resource may reside in a repository maintained by another party. In this case, the catalogue will redirect the client (HTTP status code 303, “See Other”)[5] and set the value of the `Location` header accordingly. If the catalogue cannot access the resource for any reason, then an exception message is returned instead.

8.5.3.3 Exceptions

A exception is returned if an exceptional condition arises while processing a request message. These are encapsulated in a ServiceExceptionReport; the fault codes and conditions are listed in the table below.

Table 8.600 — Exceptions associated with getExtrinsicContent

Fault code	**Fault subcode**	**Condition**
ows:Client	wrs:InvalidRequest	The request message is either invalid or is not well-formed.
ows:Server	wrs:ResourceNotFound	The resource cannot be retrieved.

---

[5] From RFC 2616: “The response to the request can be found under a different URI and SHOULD be retrieved using a GET method on that resource.”

8.5.3.4 Usage guide

The getExtrinsicContent operation MUST be bound to at least the HTTP GET method. The id element is serialized as a name-value pair within the query component of the request URI as described in the HTTP/1.1 standard [RFC 2616].

The repository item constitutes the payload of the response message. The value of the `Content-Type` entity header of the HTTP response message MUST match the value of the extrinsic object's mimeType property. If any additional encodings have been applied to the resource (e.g. compression using gzip), these must be specified by the `Content-Encoding` header.

If a fragment identifier has been specified in the request and the repository item is an XML resource, the fragment MUST conform to the XPointer framework [W3C XPointer]. A shorthand pointer may be used if it identifies an information item that is a schema-determined ID. Otherwise, a scheme-based pointer must be used to identify the desired resource fragment.

8.6 CSW-Publication interface

8.6.1 Operations provided

The `CSW-Publication` interface provides basic operations for modifying catalogue content in either a 'push' or a 'pull' manner. That is, content may be submitted directly to the catalogue or it may be harvested from some network location identified by a URL. This interface is optional for all conforming implementations—a catalogue is not required to provide a publication facility. The UML diagram in Figure 8.5.1 summarizes the interface signature.

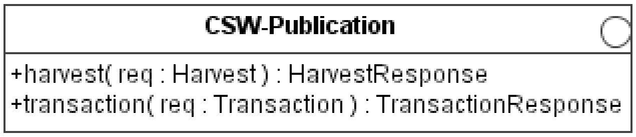
CSW-Publication
+harvest(req : Harvest) : HarvestResponse
+transaction(req : Transaction) : TransactionResponse

Figure 8.5.1 — CSW-Publication interface signature

The constituent operations are described in clause 10 of the *OpenGIS Catalogue Services 2.0 Specification* (OGC-04-021). Any extensions or restrictions introduced by the WRS profile are documented in the following subclauses.

8.6.2 transaction()

8.6.2.1 Purpose

The transaction operation allows users to insert, update, or delete registry objects. Typically this operation is subject to some kind of access control such that only authorized users may perform such actions. A transaction request message includes a collection of command statements that constitute an *atomic* unit of work: all subsidiary commands must be successfully executed or the entire transaction fails; this preserves the integrity of catalogue content, especially when a set of interrelated registry objects are being submitted.

If no exceptional conditions arise during processing of the transaction request, the service returns a TransactionResponse message that conforms to the schema in subclause 10.11 of the OpenGIS Catalogue Services 2.0 specification. In the event that the transaction fails, a `TransactionFailure` exception MUST be returned instead.

8.6.2.2 Submitting extrinsic content

When publishing extrinsic objects that include (by reference) detailed technical metadata items such as XML schemas, dataset descriptions, or style sheets (the *ancillary resource*), the entity body may be included as an attachment within a multipart message that is structured according to RFC 2387 (The MIME Multipart/Related Content-type) or RFC 2388 (Returning Values from Forms: multipart/form-data). The multipart media types are intended for compound messages that consist of several interrelated parts; such entities comprise a 'root' part plus any number of other parts or 'attachments'.

Each part in a multipart message encapsulates the actual content of an extrinsic object that is encoded as a known Internet media type[6]. The "cid" URL scheme is employed to reference other message parts, as described in RFC 2392. If a transaction request submits the content of an extrinsic object, then the value of the content/@xlink:href attribute provides a reference to the relevant message part.

8.6.2.3 Exceptions

A exception is returned if an abnormal condition arises while processing a transaction request message. These are encapsulated in a ServiceExceptionReport; the fault codes and conditions are listed in the table below.

Table 8.700 — Exceptions associated with transaction

Fault code	Fault subcode	Condition
ows:Client	wrs:InvalidRequest	The request message is either invalid or is not well-formed.
ows:Server	wrs:TransactionFailure	The transaction was not successfully completed due to the failure of one or more command statements.

---

[6] The IANA registry of media types is available at <http://www.iana.org/assignments/media-types/>.

8.6.2.4 Usage guide

The transaction operation must be bound to the HTTP POST method. The body of the request and response messages are structured in accord with the type definitions listed in the Catalogue 2.0 specification.

When specifying a fine-grained (partial) update that modifies particular properties of one or more registry objects, the value of the RecordProperty/Name element must be an XPath expression that identifies the property to be updated; the RecordProperty/Value element is of type xsd:anyType, but when updating registry objects typically only simple values are specified.

8.6.3 harvest()

8.6.3.1 Purpose

The harvest operation allows a user to request that the catalogue attempt to harvest a resource from a specified network location, thereby realizing a 'pull' model for publishing registry objects. If the catalogue successfully retrieves the resource and can handle it, then one or more corresponding registry objects are created or updated. Brief representations of all modified records are returned to the client when processing is complete.

If no exceptional conditions arise during processing of the harvest request, the service returns either a TransactionResponse message or an Acknowledgement message (if the request is processed asynchronously) that conforms to the schema in subclause 10.12 of the OpenGIS Catalogue Services 2.0 specification.

8.6.3.2 Exceptions

A exception is returned if an abnormal condition arises while processing a Harvest request message. These are encapsulated in a ServiceExceptionReport; the fault codes and conditions are listed in the table below.

Table 8.800 — Exceptions associated with harvest

Fault code	**Fault subcode**	**Condition**
ows:Client	wrs:InvalidRequest	The request message is either invalid or is not well-formed.
ows:Server	wrs:ResourceNotFound	The target resource could not be harvested for some reason
ows:Server	wrs:TransactionFailure	The transaction could not be completed.

8.6.3.3 Usage guide

The harvest operation must be bound to the HTTP POST method. The body of the request message is a <Harvest> element structured in accord with the type definition in the CSW part (clause 10) of the OpenGIS Catalogue 2.0 specification.

8.7 Query facilities

8.7.1 Spatial queries

An OGC filter expression may contain a spatial operator that specifies a query against some geometric characteristic of a registry object. The OGC filter specification [OGC 02-059] defines a number of operators that evaluate various spatial relationships, shown in Table 8.710; a conforming service need not support all of these operators.

Table 8.710 — Operators for named spatial relationships

Operator name	Must Understand
BBOX	Yes
Equals	Yes
Disjoint	Yes
Touches	Yes
Within	Yes
Overlaps	Yes
Crosses	Yes
Intersects	Yes
Contains	Yes
DWithin	No
Beyond	No

8.7.2 XPath 1.0

A WRS catalogue service can be used to query and store metadata items that are represented as XML documents; such items include GML application schemas, XSL style sheets, and dataset descriptions that conform to the ISO 19139 standard. The ability to directly query XML content is invaluable when working with such resources, as it enables “deep” discovery scenarios for users who wish to take advantage of such a sophisticated query facility.

Extrinsic registry objects may catalogue XML repository items where the value of the *mimeType* property indicates an XML media type. If such a repository item is not flagged as opaque to the catalogue (i.e., the *isOpaque* property has the value “false”), it MAY be queried using an XPath 1.0 expression according to the syntax defined in the W3C Recommendation [W3C XPath1]. See 8.4.2.2 for details.

8.7.3 Stored queries

A stored query may be invoked to perform common searches or provide a convenient ‘shortcut’ for more specialized searches. Stored queries may accept input parameters represented by Slot elements. Such predefined queries may be introduced as part of an expansion module in order to facilitate resource discovery. A StoredQuery may substitute for a csw:AbstractQuery element; the type definition for a stored query is shown in Listing 8.5.

Listing 8.5 — Type definition for a StoredQuery

```
<xsd:complexType name="StoredQueryType" id="StoredQueryType">
  <xsd:complexContent>
    <xsd:extension base="csw:AbstractQueryType">
      <xsd:sequence>
        <xsd:element name="Name" type="xsd:QName" />
        <xsd:element ref="rim:Slot"
          minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Within the context of a module definition, the value of the Slot/@slotType attribute must be included to specify the data type of the parameter (e.g., xsd:string). Listing 8.51 shows a sample definition of a stored query to list all service offers, qualified by service type if desired. Note that the <ValueList> element is empty, since the listing defines the signature of the stored query.

Listing 8.51 — Defining a StoredQuery in a module context

```
<wrs:StoredQuery
  xmlns:wrs="http://www.opengis.net/cat/wrs”
  xmlns:rim="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5”>

  <wrs:Name>listServices</wrs:Name>
  <rim:Slot
    name="serviceType”
    slotType="xsd:anyURI”>
    <rim:ValueList />
  </rim:Slot>
</wrs:StoredQuery>
```

8.7.4 Distributed search

The WRS profile supports a simple distributed search mechanism whereby synchronous or asynchronous queries may be propagated among affiliated catalogue services within a federation. Such a federation effectively enlarges the search horizon for queries submitted to a catalogue node. However, a WRS service is not required to participate in a federation. Within a federation a catalogue service plays a *mediator* role by forwarding queries to affiliated catalogues and aggregating the results. The result of a distributed search is effectively the union of a local query and one or more remote queries (Figure 8.6.1).

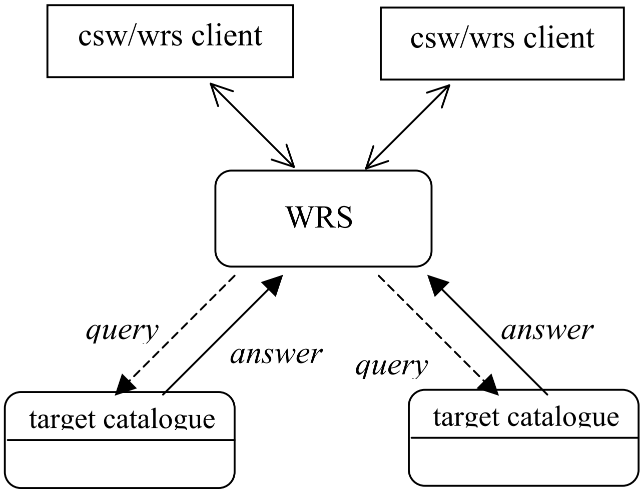

Figure 8.6.1 — WRS catalogue as mediator

A user may request that a distributed search be performed by including the <DistributedSearch> element within a GetRecords request message. If this element is not present, then the message will not be propagated. The *hopCount* attribute may be used to govern propagation behaviour by specifying the maximum number of message hops that are completed before the search is terminated. The default value is two; this value is decremented by one when the search request is received. If the *hopCount* value is zero the request is not forwarded.

The *requestId* attribute is required for all distributed request messages—this helps with loop detection. If a catalogue receives a GetRecords request message intended for distribution and it does not contain a message identifier, the catalogue shall assign a URI-based identifier before forwarding the message.

8.8 General implementation guidance

8.8.1 HTTP/1.1 message headers and response codes

The WRS catalogue profile employs HTTP/1.1 to convey request and response messages; it is a generic, stateless, application-level protocol that is widely used to exchange information on the web. The standard message headers are defined in section 14 of RFC 2616. Some of these are of particular significance in this application profile.

Any HTTP/1.1 response message containing an entity-body MUST include a Content-Type header field defining the media type of that body (RFC 2616, 7.2.1); include the charset parameter for text content. In general, this header will have the value “application/xml; charset=utf-8”. A user agent may use the Accept request header to declare a set of preferred Internet media types for the response.

The Content-Encoding entity-header may be used to indicate any additional content encodings that have been applied to the entity body, usually for the purpose of data compression or encryption. This header MUST be included if a non-identity encoding has been applied. For example, if a response message has been compressed using the standard gzip algorithm, the header will have the value “gzip”. A user agent may specify a preferred content encoding using the Accept-Encoding header. If no such request-header is included, the server MUST use the "identity" encoding.

The HTTP/1.1 protocol supports the use of conditional methods in order to curtail unnecessary network usage. A conditional header works on a particular "validator", some attribute of the resource that is tested (e.g., date of last modification, version, identifier, or some other "opaque" property). Strong validators alway uniquely identify a resource; weak ones (e.g. date of last modification, with a precision of 1s) may not always do so. Some relevant conditional request headers are summarized in Table 8.4.

Table 8.4 — Conditional HTTP requests

Request header	Validator	Description
If-Modified-Since	Last-Modified	If the requested variant has not been modified since the time specified in this field, an entity will not be returned from the server; a 304 (not modified) response instead with no message-body is sent instead.
If-Match	ETag	If none of the entity tags match, the server MUST NOT perform the requested method, and MUST return a 412 (Precondition Failed) response.

When a request is to be processed asynchronously, the response must specify status code 202 (Accepted): "The request has been accepted for processing, but the processing has not been completed" [RFC 2616]. The entity-body then provides some additional detail within the csw:Acknowledgement element.

8.8.2 Using the W3C SOAP messaging framework

The W3C SOAP 1.2 messaging framework may be used to exchange messages. When a WRS catalogue assumes the role of a SOAP node, it MUST enforce the rules governing the exchange of such messages [W3C SOAP-1]. The message payload is contained within the body part of the SOAP envelope.

A SOAP-enabled WRS catalogue MUST conform to the SOAP HTTP binding [W3C SOAP-2] and employ the two basic message exchange patterns (in either case the HTTP Content-Type header has the value “application/soap+xml”):

1. use the HTTP POST method to convey SOAP messages in the bodies of HTTP request and response messages, or
2. use the HTTP GET method in a HTTP request to return a SOAP message in the body of the HTTP response.

Compound or multipart messages may be represented in in several ways including—but not limited to—the following mechanisms:

- The primary SOAP message part and subsidiary parts may be encapsulated within a single multipart/related message, where the root part is the SOAP envelope (see <http://www.w3.org/TR/SOAP-attachments>).
- The SOAP message is exchanged using the Abstract Transmission Optimization Feature, where the xbinc:Include element refers to subsidiary parts encoded according to some media type specification (see <http://www.w3.org/TR/soap12-mtom/>).

8.9 Security considerations

8.9.1 Authentication

Catalogue search and retrieval may be performed anonymously—user authentication is not required in order to make use of any CSW discovery facility. However, some catalogue services may be required to verify the identity of a user, and in a production environment publication requests will usually be authenticated. A URI may be used to identify a supported authentication scheme; three standard authentication schemes are summarized below, but others are also allowed.

HTTP-Basic. This simple scheme is described in RFC 2617. However, since user credentials are exchanged as base64-encoded cleartext, this scheme is *not* recommended for production use.

Scheme URI: <http://www.ietf.org/rfc/rfc2617/http-basic>

HTTP-Digest. This scheme is also described in RFC 2617. It avoids the most serious flaws of HTTP-Basic and is based on a simple challenge-response paradigm; a password is never sent as cleartext—it is always transmitted as an MD5 digest of the user's password. In this way, the password cannot be determined by sniffing network traffic.

Scheme URI: <http://www.ietf.org/rfc/rfc2617/http-digest>

TLS with client certificate. This scheme requires the establishment of a secure communication channel using the TLS (Transport Layer Security) protocol, as described in RFC 2246. The host CSW server requests a public key certificate (e.g., an X.509v3 or a PGP certificate) that is then used to verify the identity of the requester.

Scheme URI: <http://www.ietf.org/rfc/rfc2246/client-cert>

8.9.2 Access control policies

In some contexts a CSW service may be required to enforce access control policies of varying scope (i.e.., service, interface, operation, resource type, resource instance, resource fragment). In such a context the default access control policy (ACP) SHOULD grant ‘read’ access to everyone but ensure that only the resource owner may perform any ‘write’ actions that modify catalogue content; this simple default policy may be overridden as required.

If one or more ACPs apply to a request, the service MUST enforce them and behave accordingly. If the user is denied access, the service MUST return an **AccessDenied** service exception (HTTP status code: 403 Forbidden); the reason for refusing the request may be included in the body of the response message.

Fine-grained access control policies can be used to restrict access to certain catalogue entries or parts of entries. In the ebRIM every registry object may be associated with exactly one access control policy that stipulates *who* is authorized to perform *what* action(s) on that object; this access control policy is an extrinsic object that generally refers to an XACML policy (or policy set) instance, but alternative policy formats are also acceptable. If no access control policy is explicitly associated with a registry object, then the default ACP applies.

An access control policy is linked to a registry object by an association instance of type name “AccessControlPolicyFor”. The source ACP need not reside in the same repository as the registry object—it could be located in a separate policy repository that exists within some security framework. The details of exactly how such policies are evaluated and authorization decisions are rendered within such a framework are not described here. However, there are a number of open standards that deal with these general issues (e.g., the OASIS specifications for XACML and SAML).

8.9.3 Maintaining an audit trail

WRS catalogues MAY maintain an audit trail for each registry object; this includes all of the events that have changed the state of the object (e.g., create, update, or delete requests). The audit trail consists of a sequence of `AuditableEvent` objects, as documented in subclause 7.2; a user may view it by submitting a getRecords() request, but typically the audit trail is subject to some level of access control (e.g. only the owner of a resource may view the audit trial).

Annex A
(normative)

Abstract test suite

A.1 General

In each Application Profile document, Annex A shall specify the Abstract Test Suite, as specified in Clause 9 and Annex A of ISO 19105. That Clause and Annex specify the ISO/TC 211 requirements for Abstract Test Suites. Examples of Abstract Test Suites are available in an annex of most ISO 191XX documents, one of the more useful is in ISO 191TBD. Note that this guidance may be more abstract than needed in an OGC Implementation Specification.

For a Catalog Implementation Profile, the Abstract Test Suite specified for the corresponding protocol binding shall be extended with more specific information. These extensions will make the Abstract Test Suite more like an Executable Test Suite, perhaps fully an Executable Test Suite.

This Abstract test suite annex is required in an OGC Application Profile, to express what a conforming implementation shall do (positively specified requirements), and what it shall not do (negatively specified requirements). Subclauses may be introduced to address different aspects:

- Assessing conformance requires consistency across the various viewpoints (i.e., clear mappings of concepts) and across the models they define; in general, the set of viewpoints should not make mutually contradictory statements. This subclause documents the mappings between views: how elements in one view are related to elements in another view

- A *reference point* identifies a behaviour or proposition that must be satisfied at a particular interaction point. A reference point may be declared as a conformance test point used to test observed behaviour; this amounts to functional or “black-box” testing that ignores implementation details and emphasizes external functionality.

- Conformance classes that bundle together a specific set of interfaces may also be defined in order to distinguish different service capability levels (e.g. a transactional catalogue, a sessional catalogue).

- Conformance testing involves testing both the capabilities and the behaviour of an implementation; it does **not** include assessment of the performance or the robustness or the reliability of an implementation. An *abstract test suite* is developed independently of any particular implementation, according to the OGC CITE framework; the abstract test cases are then transformed into executable tests that can be run on a real testing device or system. Each test case is derived from one or more reference points. In all likelihood the test suite is documented separately and referenced here.

Annex B

(normative)

XML schema for ebRIM v2.5

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
  targetNamespace="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
  xmlns:tns="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5"
  xmlns:csw="http://www.opengis.net/cat/csw"
  elementFormDefault="qualified"
  version="2.5">

  <annotation>
    <documentation xml:lang="en">
    The schema for the OASIS ebXML Registry Information Model,
    v2.5. It has been modified to import and extend
    csw:AbstractRecordType, and the signature list has been
    omitted.
    </documentation>
  </annotation>

  <import
    namespace="http://www.w3.org/XML/1998/namespace"
    schemaLocation="http://www.w3.org/2001/xml.xsd" />
  <import
    namespace="http://www.opengis.net/cat/csw"
    schemaLocation="http://schemas.opengis.net/csw/2.0.0/record" />

  <element name="Identifiable" type="tns:IdentifiableType"
    abstract="true" substitutionGroup="csw:AbstractRecord" />
  <complexType name="IdentifiableType">
    <annotation>
      <documentation xml:lang="en">
      Common base class for all types that have a public
      identifier. It extends csw:AbstractRecordType.
      </documentation>
    </annotation>
    <complexContent>
      <extension base="csw:AbstractRecordType">
        <attribute name="id" type="anyURI" use="required"/>
        <!-- home attribute is required only for remote ObjectRef -->
        <attribute name="home" type="anyURI" use="optional"/>
      </extension>
    </complexContent>
  </complexType>

  <!-- ChangeLog: added subst group; value of objectType is objectRef
in spec -->
  <element name="RegistryObject" type="tns:RegistryObjectType"
    substitutionGroup="tns:Identifiable"/>
  <complexType name="RegistryObjectType">
```

```
    <annotation>
      <documentation xml:lang="en">
        id may be empty. If specified it may be in urn:uuid format or
be in some
        arbitrary format. If id is empty registry must generate
globally unique id.
        If id is provided and in proper UUID syntax (starts with
urn:uuid:)
        registry will honour it.
        If id is provided and is not in proper UUID syntax then it is
used for
        linkage within document and is ignored by the registry. In this
case the
        registry generates a UUID for id attribute.
        id must not be null when object is being retrieved from the
registry.
      </documentation>
    </annotation>
    <complexContent>
      <extension base="tns:IdentifiableType">
        <sequence minOccurs="0" maxOccurs="1">
          <element ref="tns:Name" maxOccurs="1" minOccurs="0" />
          <element ref="tns:Description" maxOccurs="1" minOccurs="0" />
          <element ref="tns:Slot" maxOccurs="unbounded" minOccurs="0"
/>
          <element ref="tns:Classification" maxOccurs="unbounded"
minOccurs="0" />
          <element ref="tns:ExternalIdentifier" maxOccurs="unbounded"
minOccurs="0" />
        </sequence>
        <attribute name="objectType" type="anyURI" use="optional"/>
        <attribute name="status">
          <simpleType>
            <restriction base="NCName">
              <enumeration value="Submitted"/>
              <enumeration value="Approved"/>
              <enumeration value="Deprecated"/>
              <enumeration value="Withdrawn"/>
            </restriction>
          </simpleType>
        </attribute>
      </extension>
    </complexContent>
  </complexType>

  <!-- ChangeLog: added subst group -->
  <element name="RegistryEntry" type="tns:RegistryEntryType"
    substitutionGroup="tns:RegistryObject"/>
  <complexType name="RegistryEntryType">
    <complexContent>
      <extension base="tns:RegistryObjectType">
        <attribute name="expiration" type="dateTime" use="optional"/>
        <attribute default="1" name="majorVersion" type="integer"/>
        <attribute default="0" name="minorVersion" type="integer"/>
        <attribute name="stability" use="optional">
          <simpleType>
            <restriction base="NCName">
              <enumeration value="Dynamic"/>
              <enumeration value="DynamicCompatible"/>
              <enumeration value="Static"/>
            </restriction>
          </simpleType>
        </attribute>
        <attribute name="userVersion" type="tns:ShortName"
use="optional"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="Association" type="tns:AssociationType1"
    substitutionGroup="tns:RegistryObject" />
  <complexType name="AssociationType1">
    <annotation>
      <documentation xml:lang="en">
        Association is the mapping of the same named interface in
ebRIM.
        It extends RegistryObject. An Association specifies references
to
        two previously submitted registry entrys.
        The sourceObject is id of the sourceObject in association
        The targetObject is id of the targetObject in association
      </documentation>
    </annotation>
    <complexContent>
      <extension base="tns:RegistryObjectType">
        <attribute name="associationType" type="anyURI"
use="required"/>
        <attribute name="sourceObject" type="anyURI" use="required"/>
        <attribute name="targetObject" type="anyURI" use="required"/>
        <attribute name="isConfirmedBySourceOwner" type="boolean"
use="optional"/>
        <attribute name="isConfirmedByTargetOwner" type="boolean"
use="optional"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="AuditableEvent" type="tns:AuditableEventType"
    substitutionGroup="tns:RegistryObject" />
  <complexType name="AuditableEventType">
    <annotation>
      <documentation xml:lang="en">
      An Event that forms an audit trail in ebXML Registry.
      </documentation>
    </annotation>
    <complexContent>
      <extension base="tns:RegistryObjectType">
        <sequence>
          <!-- List of all objects that have been effected by this
event -->
          <element name="affectedObject" type="tns:ObjectRefListType"
            minOccurs="1" maxOccurs="1" />
        </sequence>
        <attribute name="eventType" type="tns:LongName"
use="required"/>
        <attribute name="timestamp" type="dateTime" use="required"/>
        <attribute name="user" type="anyURI" use="required"/>
        <attribute name="requestId" type="anyURI" use="required"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="Classification" type="tns:ClassificationType"
    substitutionGroup="tns:RegistryObject"/>
  <complexType name="ClassificationType">
    <annotation>
      <documentation xml:lang="en">
        Classification is the mapping of the same named interface in
ebRIM.
        It extends RegistryObject. A Classification specifies
references to
        two registry entries:
        The classifiedObject is id of the Object being classified.
        The classificationNode is id of the ClassificationNode
classying the object
      </documentation>
    </annotation>
    <complexContent>
      <extension base="tns:RegistryObjectType">
        <attribute name="classificationScheme" type="anyURI"
use="optional"/>
        <attribute name="classifiedObject" type="anyURI"
use="required"/>
        <attribute name="classificationNode" type="anyURI"
use="optional"/>
        <attribute name="nodeRepresentation" type="tns:LongName"
use="optional"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="ClassificationNode" type="tns:ClassificationNodeType"
    substitutionGroup="tns:RegistryObject"/>
  <complexType name="ClassificationNodeType">
    <annotation>
      <documentation xml:lang="en">
        ClassificationNode is the mapping of the same named interface
in ebRIM.
        It extends RegistryObject. ClassificationNode is used to submit
a Classification
        tree to the Registry. The parent attribute is the id to the
parent node; code is
        an optional code value for a ClassificationNode often defined
by an external
        taxonomy (e.g. NAICS)
      </documentation>
    </annotation>
    <complexContent>
      <extension base="tns:RegistryObjectType">
        <sequence>
          <element ref="tns:ClassificationNode" minOccurs="0"
maxOccurs="unbounded" />
        </sequence>
        <attribute name="parent" type="anyURI" use="optional"/>
        <attribute name="code" type="tns:LongName" use="optional"/>
        <attribute name="path" type="string" use="optional"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="ClassificationScheme"
type="tns:ClassificationSchemeType"
    substitutionGroup="tns:RegistryEntry"/>
  <complexType name="ClassificationSchemeType">
    <annotation>
      <documentation xml:lang="en">
        ClassificationScheme is the mapping of the same named interface
in ebRIM.
        It extends RegistryEntry.
      </documentation>
    </annotation>
    <complexContent>
      <extension base="tns:RegistryEntryType">
        <sequence>
          <element ref="tns:ClassificationNode" minOccurs="0"
maxOccurs="unbounded" />
        </sequence>
        <attribute name="isInternal" type="boolean" use="required"/>
        <attribute name="nodeType" use="required">
          <simpleType>
            <restriction base="NCName">
              <enumeration value="UniqueCode"/>
              <enumeration value="EmbeddedPath"/>
              <enumeration value="NonUniqueCode"/>
            </restriction>
          </simpleType>
        </attribute>
      </extension>
    </complexContent>
  </complexType>

  <element name="ExternalIdentifier" type="tns:ExternalIdentifierType"
    substitutionGroup="tns:RegistryObject"/>
  <complexType name="ExternalIdentifierType">
    <annotation>
      <documentation xml:lang="en">
        ExternalIdentifier is the mapping of the same named interface
in ebRIM.
        It extends RegistryObject.
      </documentation>
    </annotation>
    <complexContent>
      <extension base="tns:RegistryObjectType">
        <attribute name="registryObject" type="anyURI" use="optional"/>
        <attribute name="identificationScheme" type="anyURI"
use="required"/>
        <attribute name="value" type="tns:LongName" use="required"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="ExternalLink" type="tns:ExternalLinkType"
```

```
    substitutionGroup="tns:RegistryObject"/>
  <complexType name="ExternalLinkType">
    <annotation>
      <documentation xml:lang="en">
        ExternalLink is the mapping of the same named interface in
ebRIM.
        It extends RegistryObject.
      </documentation>
    </annotation>
    <complexContent>
      <extension base="tns:RegistryObjectType">
        <attribute name="externalURI" type="anyURI" use="required"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="ExtrinsicObject" type="tns:ExtrinsicObjectType"
    substitutionGroup="tns:RegistryEntry"/>
  <complexType name="ExtrinsicObjectType">
    <annotation>
      <documentation xml:lang="en">
        ExtrinsicObject are attributes from the ExtrinsicObject
interface in ebRIM.
        It inherits RegistryEntryAttributes
      </documentation>
    </annotation>
    <complexContent>
      <extension base="tns:RegistryEntryType">
        <attribute name="mimeType" type="tns:LongName"
default="application/octet-stream" />
        <attribute name="isOpaque" type="boolean" default="false" />
      </extension>
    </complexContent>
  </complexType>

  <element name="Organization" type="tns:OrganizationType"
    substitutionGroup="tns:RegistryObject"/>
  <complexType name="OrganizationType">
    <annotation>
      <documentation xml:lang="en">
      Mapping of the same named interface in ebRIM.
      </documentation>
    </annotation>
    <complexContent>
      <extension base="tns:RegistryObjectType">
        <sequence minOccurs="1" maxOccurs="1">
          <element ref="tns:Address" minOccurs="1" maxOccurs="1" />
          <element ref="tns:TelephoneNumber" minOccurs="1"
maxOccurs="unbounded" />
          <element ref="tns:EmailAddress" minOccurs="0"
maxOccurs="unbounded" />
        </sequence>
        <attribute name="parent" type="anyURI"/>
        <attribute name="primaryContact" type="anyURI" use="required"/>
      </extension>
    </complexContent>
  </complexType>
  <element name="RegistryPackage" type="tns:RegistryPackageType"
    substitutionGroup="tns:RegistryEntry" />
  <complexType name="RegistryPackageType">
    <annotation>
      <documentation xml:lang="en">
        RegistryPackage is the mapping of the same named interface in
ebRIM.
        It extends RegistryEntry. A RegistryPackage is a named
collection of
        objects.
      </documentation>
    </annotation>
    <complexContent>
      <extension base="tns:RegistryEntryType">
        <sequence>
          <element ref="tns:RegistryObjectList" minOccurs="0"
maxOccurs="1" />
        </sequence>
      </extension>
    </complexContent>
  </complexType>

  <element name="Service" type="tns:ServiceType"
    substitutionGroup="tns:RegistryEntry"/>
  <complexType name="ServiceType">
    <complexContent>
      <extension base="tns:RegistryEntryType">
        <sequence>
          <element ref="tns:ServiceBinding" minOccurs="0"
maxOccurs="unbounded" />
        </sequence>
      </extension>
    </complexContent>
  </complexType>

  <element name="ServiceBinding" type="tns:ServiceBindingType"
    substitutionGroup="tns:RegistryObject"/>
  <complexType name="ServiceBindingType">
    <complexContent>
      <extension base="tns:RegistryObjectType">
        <sequence>
          <element maxOccurs="unbounded" minOccurs="0"
ref="tns:SpecificationLink"/>
        </sequence>
        <attribute name="service" type="anyURI" use="optional"/>
        <attribute name="accessURI" type="anyURI" use="optional"/>
        <attribute name="targetBinding" type="anyURI" use="optional"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="SpecificationLink" type="tns:SpecificationLinkType"
    substitutionGroup="tns:RegistryObject"/>
  <complexType name="SpecificationLinkType">
    <complexContent>
      <extension base="tns:RegistryObjectType">
        <sequence maxOccurs="1" minOccurs="0">
```

```
          <element ref="tns:UsageDescription" minOccurs="0"
maxOccurs="1" />
          <element ref="tns:UsageParameter" minOccurs="0"
maxOccurs="unbounded" />
        </sequence>
        <attribute name="serviceBinding" type="anyURI" use="optional"/>
        <attribute name="specificationObject" type="anyURI"
use="required"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="UsageDescription" type="tns:InternationalStringType"/>
  <element name="UsageParameter" type="tns:FreeFormText"/>

  <element name="User" type="tns:UserType"
    substitutionGroup="tns:RegistryObject"/>
  <complexType name="UserType">
    <annotation>
      <documentation xml:lang="en">
      Mapping of the same named interface in ebRIM.
      </documentation>
    </annotation>
    <complexContent>
      <extension base="tns:RegistryObjectType">
        <sequence>
          <element ref="tns:Address" minOccurs="1"
maxOccurs="unbounded" />
          <element ref="tns:PersonName" />
          <element ref="tns:TelephoneNumber" minOccurs="1"
maxOccurs="unbounded" />
          <element ref="tns:EmailAddress" minOccurs="1"
maxOccurs="unbounded" />
        </sequence>
        <attribute name="url" type="anyURI" use="optional"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="Registry" type="tns:RegistryType"
    substitutionGroup="tns:RegistryEntry"/>
  <complexType name="RegistryType">
    <annotation>
      <documentation xml:lang="en">
      Mapping of the same named interface in ebRIM.
      </documentation>
    </annotation>
    <complexContent>
      <extension base="tns:RegistryEntryType">
        <attribute name="operator" type="anyURI" use="required"/>
        <attribute name="specificationVersion" type="string"
use="required"/>
        <attribute name="replicationSyncLatency" type="duration"
default="P1D" />
        <attribute name="catalogingLatency" type="duration"
default="P1D" />
        <!-- Optional features supported -->
        <attribute name="sqlQuerySupported" type="boolean"
default="false" />
        <attribute name="eventNotificationSupported" type="boolean"
default="false" />
        <attribute name="objectReplicationSupported" type="boolean"
default="false" />
        <attribute name="objectRelocationSupported" type="boolean"
default="false" />
      </extension>
    </complexContent>
  </complexType>

  <element name="Federation" type="tns:FederationType"
    substitutionGroup="tns:RegistryEntry" />
  <element name="Members" type="tns:ObjectRefListType" />
  <complexType name="FederationType">
    <annotation>
      <documentation xml:lang="en">
      Mapping of the same named interface in ebRIM.
      </documentation>
    </annotation>
    <complexContent>
      <extension base="tns:RegistryEntryType">
        <sequence>
          <element ref="tns:Members" />
        </sequence>
        <attribute name="replicationSyncLatency" type="duration"
use="required"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="Subscription" type="tns:SubscriptionType"
    substitutionGroup="tns:RegistryObject"/>
  <complexType name="SubscriptionType">
    <annotation>
      <documentation xml:lang="en">
      A Subscription for specified Events in an ebXML V3+ registry.
      </documentation>
    </annotation>
    <complexContent>
      <extension base="tns:RegistryObjectType">
        <sequence>
          <element ref="tns:Action" minOccurs="1"
maxOccurs="unbounded"/>
        </sequence>
        <!-- Ref to a AdhocQueryType instance -->
        <attribute name="selector" type="anyURI" use="required"/>
        <attribute name="startDate" type="dateTime" use="optional"/>
        <attribute name="endDate" type="dateTime" use="optional"/>
        <attribute name="notificationInterval" type="duration"
use="optional"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="Action" type="tns:ActionType"/>
  <complexType abstract="true" name="ActionType">
```

```
    <annotation>
      <documentation>Abstract Base type for all types of
Actions.</documentation>
    </annotation>
  </complexType>

  <element name="NotifyAction" type="tns:NotifyActionType"
    substitutionGroup="tns:Action"/>
  <complexType name="NotifyActionType">
    <annotation>
      <documentation xml:lang="en">
      Abstract Base type for all types of Notify Actions
      </documentation>
    </annotation>
    <complexContent>
      <extension base="tns:ActionType">
        <attribute name="notificationOption" default="ObjectRefs">
          <simpleType>
            <restriction base="NCName">
              <enumeration value="ObjectRefs"/>
              <enumeration value="Objects"/>
            </restriction>
          </simpleType>
        </attribute>
        <attribute name="endPoint" type="anyURI" use="required"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="Slot" type="tns:SlotType1"/>
  <complexType name="SlotType1">
    <sequence>
      <element ref="tns:ValueList" />
    </sequence>
    <attribute name="name" type="tns:LongName" use="required"/>
    <attribute name="slotType" type="tns:LongName" use="optional"/>
  </complexType>

  <element name="ValueList" type="tns:ValueListType"/>
  <complexType name="ValueListType">
    <sequence minOccurs="0" maxOccurs="unbounded">
      <element ref="tns:Value" />
    </sequence>
  </complexType>

  <element name="Value" type="tns:LongName"/>

  <element name="SlotList" type="tns:SlotListType"/>
  <complexType name="SlotListType">
    <sequence>
      <element ref="tns:Slot" minOccurs="0" maxOccurs="unbounded" />
    </sequence>
  </complexType>

  <element name="PersonName" type="tns:PersonNameType"/>
  <complexType name="PersonNameType">
    <annotation>
      <documentation xml:lang="en">
      Mapping of the same named interface in ebRIM.
      </documentation>
    </annotation>
    <sequence minOccurs="0" maxOccurs="unbounded">
      <element ref="tns:Slot" />
    </sequence>
    <attribute name="firstName" type="tns:ShortName" use="optional"/>
    <attribute name="middleName" type="tns:ShortName" use="optional"/>
    <attribute name="lastName" type="tns:ShortName" use="optional"/>
  </complexType>

  <element name="EmailAddress" type="tns:EmailAddressType"/>
  <complexType name="EmailAddressType">
    <annotation>
      <documentation xml:lang="en">
      Mapping of the same named interface in ebRIM.
      </documentation>
    </annotation>
    <sequence minOccurs="0" maxOccurs="unbounded">
      <element ref="tns:Slot"/>
    </sequence>
    <attribute name="address" type="tns:ShortName" use="required"/>
    <attribute name="type" type="tns:String32" use="optional"/>
  </complexType>

  <element name="Address" type="tns:PostalAddressType" />
  <element name="PostalAddress" type="tns:PostalAddressType" />
  <complexType name="PostalAddressType">
    <annotation>
      <documentation xml:lang="en">
      Mapping of the same named interface in ebRIM.
      </documentation>
    </annotation>
    <sequence minOccurs="0" maxOccurs="unbounded">
      <element ref="tns:Slot"/>
    </sequence>
    <attribute name="city" type="tns:ShortName" use="optional"/>
    <attribute name="country" type="tns:ShortName" use="optional"/>
    <attribute name="postalCode" type="tns:ShortName" use="optional"/>
    <attribute name="stateOrProvince" type="tns:ShortName"
use="optional"/>
    <attribute name="street" type="tns:ShortName" use="optional"/>
    <attribute name="streetNumber" type="tns:String32" use="optional"/>
  </complexType>

  <element name="Name" type="tns:InternationalStringType"/>
  <element name="Description" type="tns:InternationalStringType"/>
  <element name="InternationalString"
type="tns:InternationalStringType"/>
  <complexType name="InternationalStringType">
    <sequence minOccurs="0" maxOccurs="unbounded">
      <element ref="tns:LocalizedString"/>
    </sequence>
  </complexType>

  <element name="LocalizedString" type="tns:LocalizedStringType"/>
  <complexType name="LocalizedStringType">
    <attribute ref="xml:lang" default="en-US"/>
    <attribute name="charset"  type="string" default="UTF-8" />
    <attribute name="value" type="tns:FreeFormText" use="required"/>
  </complexType>

  <element name="TelephoneNumber" type="tns:TelephoneNumberType"/>
  <element name="FaxNumber" type="tns:TelephoneNumberType"/>
  <element name="MobileTelephoneNumber"
type="tns:TelephoneNumberType"/>
  <element name="PagerNumber" type="tns:TelephoneNumberType"/>
  <complexType name="TelephoneNumberType">
    <annotation>
      <documentation xml:lang="en">
      TelephoneNumber is the mapping of the same named interface in
ebRIM.
      </documentation>
    </annotation>
    <attribute name="areaCode" type="tns:String8" use="optional"/>
    <attribute name="countryCode" type="tns:String8" use="optional"/>
    <attribute name="extension" type="tns:String8" use="optional"/>
    <attribute name="number" type="tns:String16" use="optional"/>
    <attribute name="phoneType" type="tns:String32" use="optional"/>
    <attribute name="url" type="anyURI" use="optional"/>
  </complexType>

  <complexType name="TelephoneNumberListType">
    <sequence>
      <element ref="tns:TelephoneNumber" minOccurs="0"
maxOccurs="unbounded" />
    </sequence>
  </complexType>

  <complexType abstract="true" name="AdhocQueryType">
    <annotation>
      <documentation xml:lang="en">The common base type for all types
of Adh hoc queries.</documentation>
    </annotation>
    <complexContent>
      <extension base="tns:RegistryObjectType"/>
    </complexContent>
  </complexType>

  <element name="ObjectRef" type="tns:ObjectRefType"
    substitutionGroup="tns:Identifiable"/>
  <complexType name="ObjectRefType">
    <annotation>
      <documentation xml:lang="en">
        Use to reference an Object by its id.
        Specifies the id attribute of the object as its id attribute.
        id attribute in ObjectAttributes is exactly the same syntax and
semantics as
        id attribute in RegistryObject.
      </documentation>
    </annotation>
    <complexContent>
      <extension base="tns:IdentifiableType">
        <!-- When true and is a remote ObjectRef then the registry must
create a replica
        for this ObjectRef -->
        <attribute name="createReplica" type="boolean" default="false"
/>
      </extension>
    </complexContent>
  </complexType>

  <element name="ObjectRefList" type="tns:ObjectRefListType" />
  <complexType name="ObjectRefListType">
    <sequence minOccurs="0" maxOccurs="unbounded">
      <element ref="tns:ObjectRef"/>
    </sequence>
  </complexType>

  <!-- change to IdentifiableList? -->
  <element name="RegistryObjectList"
type="tns:RegistryObjectListType"/>
  <complexType name="RegistryObjectListType">
    <sequence>
      <element ref="tns:Identifiable" minOccurs="0"
maxOccurs="unbounded" />
    </sequence>
  </complexType>

  <!-- define simple data types -->
  <simpleType name="String4">
    <restriction base="string">
      <maxLength value="4"/>
    </restriction>
  </simpleType>
  <simpleType name="String8">
    <restriction base="string">
      <maxLength value="8"/>
    </restriction>
  </simpleType>
  <simpleType name="String16">
    <restriction base="string">
      <maxLength value="16"/>
    </restriction>
  </simpleType>
  <simpleType name="String32">
    <restriction base="string">
      <maxLength value="32"/>
    </restriction>
  </simpleType>
  <simpleType name="ShortName">
    <restriction base="string">
      <maxLength value="64"/>
    </restriction>
  </simpleType>
  <simpleType name="LongName">
    <restriction base="string">
      <maxLength value="128"/>
    </restriction>
  </simpleType>
  <simpleType name="FreeFormText">
    <restriction base="string">
      <maxLength value="256"/>
    </restriction>
  </simpleType>
</schema>
```

Annex C
(normative)

WRS extensions to ebRIM

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema id="wrs-rim"
  targetNamespace="http://www.opengis.net/cat/wrs"
  xmlns:wrs="http://www.opengis.net/cat/wrs"
  xmlns:csw="http://www.opengis.net/cat/csw"
  xmlns:rim="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  version="0.9.0">

  <xsd:annotation>
    <xsd:appinfo>
      <dc:identifier xmlns:dc="http://purl.org/dc/elements/1.1/">
      http://schemas.opengis.net/wrs/0.9.0/rim
      </dc:identifier>
      <dct:modified xmlns:dct="http://purl.org/dc/terms/">
      2004-05-24
      </dct:modified>
    </xsd:appinfo>
    <xsd:documentation xml:lang="en">
    WRS extensions to the ebXML Registry Information Model, v2.5.
    </xsd:documentation>
  </xsd:annotation>

  <!-- import the ebRIM-2.5 schema -->
  <xsd:import
    namespace="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5"
    schemaLocation="http://www.oasis-
open.org/committees/regrep/documents/2.5/schema/rim.xsd" />
  <!-- import csw:AbstractQuery -->
  <xsd:import namespace="http://www.opengis.net/cat/csw"
    schemaLocation="http://schemas.opengis.net/csw/2.0.0/discovery" />
  <!-- import XLink attributes -->
  <xsd:import
    namespace="http://www.w3.org/1999/xlink"
    schemaLocation="http://schemas.opengis.net/gml/2.1.2/xlinks.xsd" />

  <xsd:element name="WRSExtrinsicObject" id="WRSExtrinsicObject"
    type="wrs:WRSExtrinsicObjectType"
    substitutionGroup="rim:ExtrinsicObject" />
  <xsd:complexType name="WRSExtrinsicObjectType"
id="WRSExtrinsicObjectType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
      Extends ExtrinsicObjectType to include a content element that
provides
      a reference to a representation of the extrinsic content
available in a
```

```
      repository; the repository may be maintained by a third-party
provider.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="rim:ExtrinsicObjectType">
        <xsd:sequence>
          <xsd:element name="content" type="wrs:SimpleLinkType" />
   </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="SimpleLinkType" id="SimpleLinkType">
    <xsd:annotation>
      <xsd:documentation>
        The xlink:href property provides a reference to the content
associated
   with an extrinsic object. The value may be an absolute or a relative
   URI as defined in RFC 2396. If the URI specifies a resource located
in
   a repository maintained by another party, then the catalogue must
   perform an HTTP redirect (status code 303) and set the value of the
   Location header accordingly.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:attributeGroup ref="xlink:simpleLink"/>
  </xsd:complexType>

  <xsd:element name="Geometry" id="Geometry"
    type="wrs:GeometryType"
    substitutionGroup="wrs:WRSExtrinsicObject" />
  <xsd:complexType name="GeometryType" id="GeometryType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
      Extends WRSExtrinsicObjectType to include the basic properties of
a
      geometry instance as defined in the OGC simple features model
(99-049).
      The repositoryItem property references an encoding of the
geometry
      instance.
      dimension    - inherent dimension of the geometry instance
      geometryType - any concrete GML 3 geometry type
      srid         - id of the spatial reference system for this
instance
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="wrs:WRSExtrinsicObjectType">
        <xsd:sequence>
          <xsd:element name="dimension" type="xsd:positiveInteger"
            minOccurs="0" />
          <xsd:element name="geometryType" type="xsd:anyURI" />
          <xsd:element name="srid" type="xsd:anyURI" minOccurs="0" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="ApplicationModule" id="ApplicationModule"
    type="wrs:ApplicationModuleType"
    substitutionGroup="rim:RegistryPackage"/>
  <xsd:complexType name="ApplicationModuleType"
id="ApplicationModuleType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
      Defines an expansion pack that bundles extensions for handling
      domain-specific resources; it may include ClassificationScheme,
      ClassificationNode, Association, XMLSchema, WRS query extensions
(e.g.
      stored queries), plus constraints on the use of slots.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="rim:RegistryPackageType">
        <xsd:sequence>
          <xsd:element ref="csw:AbstractQuery"
            minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

Annex D
(normative)

WRS extensions to CSW

Listing D.1 — WRS-Discovery messages

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema id="wrs-retrieval"
  targetNamespace="http://www.opengis.net/cat/wrs"
  xmlns:csw="http://www.opengis.net/cat/csw"
  xmlns:wrs="http://www.opengis.net/cat/wrs"
  xmlns:rim="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  version="0.9.0">

  <xsd:annotation>
    <xsd:appinfo>
      <dc:identifier xmlns:dc="http://purl.org/dc/elements/1.1/">
      http://schemas.opengis.net/wrs/0.9.0/retrieval
      </dc:identifier>
      <dct:modified xmlns:dct="http://purl.org/dc/terms/">
      2004-05-24
      </dct:modified>
    </xsd:appinfo>
    <xsd:documentation xml:lang="en">
    This schema defines additional message content for WRS that extends
the
    information retrieval capabilities of CSW-based catalogues.
    </xsd:documentation>
  </xsd:annotation>

  <!-- include WRS extensions to ebRIM -->
  <xsd:include
    schemaLocation="http://schemas.opengis.net/wrs/0.9.0/rim" />
  <!-- import CSW discovery message elements -->
  <xsd:import
    namespace="http://www.opengis.net/cat/csw"
    schemaLocation="http://schemas.opengis.net/csw/2.0.0/discovery" />
  <!-- import ebRIM 2.5 components -->
  <xsd:import
    namespace="urn:oasis:names:tc:ebxml-regrep:rim:xsd:2.5"
    schemaLocation="http://www.oasis-
open.org/committees/regrep/documents/2.5/schema/rim.xsd" />
  <!-- import sortBy element -->
  <xsd:import
    namespace="http://www.opengis.net/ogc"
    schemaLocation="http://schemas.opengis.net/filter/1.0.20/sort.xsd"
/>

  <xsd:element name="XPathQuery" type="wrs:XPathQueryType"
id="XPathQuery"
```

```
     substitutionGroup="csw:AbstractQuery" />
  <xsd:complexType name="XPathQueryType" id="XPathQueryType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
      Evaluates XPath expressions against the content of extrinsic
objects. The
      result set is populated with those objects for which there is a
non-empty
      XPath result.
      context - the context node for the query
      xpath   - the path expression to be evaluated
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="csw:AbstractQueryType">
        <xsd:sequence>
     <xsd:element name="elementSet" type="csw:ElementSetType"
       minOccurs="0" default="brief" />
     <xsd:element name="context" type="xsd:string" />
     <xsd:element name="xpath" type="xsd:string" />
        </xsd:sequence>
   <xsd:attribute name="version" type="xsd:string" use="required" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="StoredQuery" id="StoredQuery"
type="wrs:StoredQueryType"
    substitutionGroup="csw:AbstractQuery"  />
  <xsd:complexType name="StoredQueryType" id="StoredQueryType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
      A stored query may be invoked by name to perform common searches
or to
      provide a convenient way of executing complex queries. Stored
queries
      may accept input parameters represented as instances of rim:Slot.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="csw:AbstractQueryType">
        <xsd:sequence>
     <xsd:element name="Name" type="xsd:QName" />
          <xsd:element ref="rim:Slot" minOccurs="0"
maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="PresentResults" id="PresentResults"
    type="wrs:PresentResultsType" />
  <xsd:complexType name="PresentResultsType" id="PresentResultsType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
      Allows a client to retrieve the members of a specified result set
      (referenced by URI) in a particular presentation format. A
sorting
```

```
      criterion that specifies a property to sort on may be included
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="resultSetId" type="xsd:anyURI" />
      <xsd:element ref="ogc:SortBy" minOccurs="0" />
    </xsd:sequence>
    <xsd:attribute name="elementSet" type="csw:ElementSetType"
use="optional" />
    <xsd:attributeGroup ref="csw:BasicRetrievalOptions" />
  </xsd:complexType>

  <xsd:element name="PresentResultsResponse"
id="PresentResultsResponse"
    type="csw:SearchResultsType">
     <xsd:annotation>
      <xsd:documentation>
      Representations of result set members must conform to the
requested
      output schema.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>

  <xsd:element name="GetExtrinsicContent" id="GetExtrinsicContent"
    type="wrs:GetExtrinsicContentType" />
  <xsd:complexType name="GetExtrinsicContentType"
id="GetExtrinsicContentType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
      Requests the content associated with an extrinsic object, or a
fragment
      thereof if the service can resolve the fragment identifier. The
item is
      returned as the entity-body in the HTTP response.

      id - object identifier (a URI)
      fragment - a fragment identifier, the syntax and semantics of
which is
        dependent upon the media type of the repository item. For XML
resources
        the W3C XPointer framework MUST be used.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="id" type="xsd:anyURI" />
      <xsd:element name="fragment" type="xsd:string" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>

</xsd:schema>
```

Annex E
(normative)

WRS capabilities: additional information

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema id="wrs-extra"
  targetNamespace="http://www.opengis.net/cat/wrs"
  xmlns:wrs="http://www.opengis.net/cat/wrs"
  xmlns:csw="http://www.opengis.net/cat/csw"
  xmlns:ows="http://www.opengis.net/ows"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  version="0.9.0">

  <xsd:annotation>
    <xsd:appinfo>
      <dc:identifier xmlns:dc="http://purl.org/dc/elements/1.1/">
      http://schemas.opengis.net/wrs/0.9.0/wrs-extra
      </dc:identifier>
      <dct:modified xmlns:dct="http://purl.org/dc/terms/">
      2004-05-31
      </dct:modified>
    </xsd:appinfo>
    <xsd:documentation xml:lang="en">
    This schema defines various extra WRS-specific information items
    that extend common OWS elements.
    </xsd:documentation>
  </xsd:annotation>

  <!-- import CSW capabilities elements -->
  <xsd:import
    namespace="http://www.opengis.net/cat/csw"
    schemaLocation="http://schemas.opengis.net/csw/2.0.0/discovery" />
  <!-- import OWS exception elements -->
  <xsd:import
    namespace="http://www.opengis.net/ows"
    schemaLocation="http://schemas.opengis.net/ows/0.3.0/exceptions" />

  <xsd:element name="WRSException" id="WRSException"
    type="wrs:WRSExceptionType"
    substitutionGroup="ows:Exception"/>
  <xsd:complexType name="WRSExceptionType" id="WRSExceptionType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
      Extends ows:ExceptionType to provide additional information about
      the service exception.
      subCode - refines the more general exceptionCode
      detail  - an optional diagnostic message.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="ows:ExceptionType">
        <xsd:sequence>
```

```
     <xsd:element name="Detail" type="xsd:anyType" minOccurs="0" />
        </xsd:sequence>
   <xsd:attribute name="subCode" type="xsd:QName" use="required" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="Capabilities" id="Capabilities"
    type="wrs:WRSCapabilitiesType"
    substitutionGroup="csw:Capabilities"/>
  <xsd:complexType name="WRSCapabilitiesType" id="WRSCapabilitiesType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
      Extends csw:CapabilitiesType to provide additional information
about
      WRS-specific capabilities.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="csw:CapabilitiesType">
        <xsd:sequence>
     <xsd:element name="ExtraServiceInfo"
type="wrs:ExtraServiceInfoType" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="ExtraServiceInfoType"
id="ExtraServiceInfoType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
      WRS-specific metadata about service capabilities.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="feature" type="wrs:ServiceFeatureType"
        minOccurs="0" maxOccurs="unbounded" />
      <xsd:element name="property" type="wrs:ServicePropertyType"
        minOccurs="0" maxOccurs="unbounded" />
      <xsd:element name="modules" type="wrs:ModulesType" />
      <xsd:element name="data-formats" type="wrs:DataFormatsType" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="ServiceFeatureType" id="ServiceFeatureType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
      Indicates whether the specified service feature is supported.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="name" type="xsd:anyURI" use="required" />
    <xsd:attribute name="isSupported" type="xsd:boolean" use="required"
/>
  </xsd:complexType>

  <xsd:complexType name="ServicePropertyType" id="ServicePropertyType">
    <xsd:annotation>
```

```
      <xsd:documentation xml:lang="en">
      Provides a value for the specified service property.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="name" type="xsd:anyURI" use="required" />
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <xsd:complexType name="ModulesType" id="ModulesType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
      Provides a list of supported WRS modules. The Standard module is
required.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="module" type="wrs:ModuleType"
        minOccurs="1" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="ModuleType" id="ModuleType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
      Identifies a module that bundles WRS extensions.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="name" type="xsd:anyURI" use="required" />
    <xsd:attribute name="version" type="xsd:boolean" use="required" />
  </xsd:complexType>

  <xsd:complexType name="DataFormatsType" id="DataFormatsType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
      Provides a list of supported data formats for catalogue content.
At least
      two are required: the csw:Record format plus the default ebRIM
      representation. While any content type may be referenced, only
resources
      conforming to the specified content types may be submitted as
extrinsic
      content for publication.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="data-format" type="wrs:DataFormatType"
        minOccurs="2" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="DataFormatType" id="DataFormatType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
      Describes a supported representation of catalogue content.
      contentType - the Internet media type of the resource
      schema      - identifies the formal schema, if applicable
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="schema" type="wrs:SchemaType" minOccurs="0" />
    </xsd:sequence>
    <xsd:attribute name="contentType" type="xsd:string" use="required"
/>
  </xsd:complexType>

  <xsd:complexType name="SchemaType" id="SchemaType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
      Provides a reference to the relevant schema for the data format.
      id - identifier of the registry object for the schema
      schemaLanguage - indicates the schema language
      </xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="id" type="xsd:anyURI" use="required" />
    <xsd:attribute name="schemaLanguage" type="xsd:anyURI"
use="required" />
  </xsd:complexType>

</xsd:schema>
```

Annex F
(normative)

The Standard WRS module

Extrinsic object types

Table F.1 — New extrinsic object types

Extrinsic object type	Description
ServiceProfile	Describes the capabilities of a service (from the OWL-S service ontology).
ServiceModel	Describes the computational characteristics of a service (from the OWL-S service ontology).
ServiceGrounding	Describes the protocol bindings and network endpoints used to access a service implementation (from the OWL-S service ontology).
Dataset	Description of a geographic data set.
Schema	A formal description of a structured information resource, expressed using some schema language (e.g., W3C XML Schema, RELAX NG, UML, ASN.1).
Stylesheet	A presentation or portrayal resource that defines styling rules (e.g., XSLT, XSL-FO, CSS-2, DSSSL).
AccessControlPolicy	Defines access control rules for establishing whether a principal is allowed to perform a requested action.
Document	Documentation of all kinds (e.g. specifications, manuals, reports).
Dictionary	A compendium of item definitions (e.g., units of measurement, vocabulary terms).
Annotation	A resource that interprets, explains, clarifies, defines, or otherwise supplements a related resource.
Image	A symbolic visual resource other than text (e.g., diagrams, photographs, drawings, animations).

Listing F.1 — Schema for the Standard module

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema id="mod-standard"
  targetNamespace="http://www.opengis.net/cat/wrs"
  xmlns:wrs="http://www.opengis.net/cat/wrs"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  version="0.9.0">

  <xsd:annotation>
    <xsd:appinfo>
      <dc:identifier xmlns:dc="http://purl.org/dc/elements/1.1/">
      http://schemas.opengis.net/wrs/0.9.0/modules/standard
      </dc:identifier>
      <dct:modified xmlns:dct="http://purl.org/dc/terms/">
      2004-05-25
      </dct:modified>
    </xsd:appinfo>
    <xsd:documentation xml:lang="en">
    Declares new extrinsic object types in the Standard module.
    </xsd:documentation>
  </xsd:annotation>

  <!-- include WRS extensions to ebRIM -->
  <xsd:include
    schemaLocation="http://schemas.opengis.net/wrs/0.9.0/rim"/>

  <xsd:element name="ServiceProfile" id="ServiceProfile"
    type="wrs:WRSExtrinsicObjectType"
    substitutionGroup="wrs:WRSExtrinsicObject">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
      Describes the capabilities of a service (from the OWL-S
      service ontology).
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="ServiceModel" id="ServiceModel"
    type="wrs:WRSExtrinsicObjectType"
    substitutionGroup="wrs:WRSExtrinsicObject">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
      Describes the computational characteristics of a service
      (from the OWL-S service ontology).
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="ServiceGrounding" id="ServiceGrounding"
    type="wrs:WRSExtrinsicObjectType"
    substitutionGroup="wrs:WRSExtrinsicObject">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
      Describes the protocol bindings and network endpoints used to
      access a service implementation (from the OWL-S service
      ontology).
      </xsd:documentation>
    </xsd:annotation>
```

```
</xsd:element>
<xsd:element name="Dataset" id="Dataset"
  type="wrs:WRSExtrinsicObjectType"
  substitutionGroup="wrs:WRSExtrinsicObject">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
    Description of a geographic data set.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="Schema" id="Schema"
  type="wrs:WRSExtrinsicObjectType"
  substitutionGroup="wrs:WRSExtrinsicObject">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
    A formal description of a structured information resource,
    expressed using some schema language (e.g., W3C XML Schema,
    ISO RELAX, UML, ASN.1).
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="Stylesheet" id="Stylesheet"
  type="wrs:WRSExtrinsicObjectType"
  substitutionGroup="wrs:WRSExtrinsicObject">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
    A presentation resource that defines styling rules (e.g., XSLT,
    CSS, DSSSL).
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="AccessControlPolicy" id="AccessControlPolicy"
  type="wrs:WRSExtrinsicObjectType"
  substitutionGroup="wrs:WRSExtrinsicObject">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
    Defines access control rules for establishing whether a principal
    is allowed to perform a requested action.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="Document" id="Document"
  type="wrs:WRSExtrinsicObjectType"
  substitutionGroup="wrs:WRSExtrinsicObject">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
    Documentation of all kinds (e.g. specifications,
    manuals, reports).
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="Dictionary" id="Dictionary"
  type="wrs:WRSExtrinsicObjectType"
  substitutionGroup="wrs:WRSExtrinsicObject">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
    A compendium of item definitions (e.g., units of measurement,
    vocabulary terms).
```

```
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="Annotation" id="Annotation"
    type="wrs:WRSExtrinsicObjectType"
    substitutionGroup="wrs:WRSExtrinsicObject">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
      A resource that interprets, explains, clarifies, defines,
      or otherwise supplements a related resource.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="Image" id="Image"
    type="wrs:WRSExtrinsicObjectType"
    substitutionGroup="wrs:WRSExtrinsicObject">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
      A symbolic visual resource other than text.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
</xsd:schema>
```

classification schemes

- ISO 19119 service types
- ISO 3166-1 (country codes)
- DIGEST v2.1 feature categories (from Part 4, FACC)

classification nodes

The object types listed in Table F.1 extend the canonical object types scheme under the `WRSExtrinsicObject` node.

associations

Table F.2 — New association types

Association type	Source object type	Destination object type(s)
presents (DAML-S)	Service	ServiceProfile
describedBy (DAML-S)	Service	ServiceModel
supports (DAML-S)	Service	ServiceGrounding
operatesOn (ISO 19119)	Service	Dataset
annotatedBy	RegistryObject	Annotation

Association type	Source object type	Destination object type(s)
applicationSchemaInfo (ISO 19115)	Dataset	Schema

slots

Table F.3 — Slots

Name	slotType	Parent type(s)
modified	http://purl.org/dc/terms/modified	RegistryObject
subject	http://purl.org/dc/elements/1.1/subject	RegistryObject
language	http://purl.org/dc/elements/1.1/language	RegistryObject
fragment	http://www.w3.org/TR/xptr-framework/	Association,Classification
role	http://www.opengis.net/cat/wrs/role	Classification
temporal	http://purl.org/dc/terms/temporal	RegistryObject

stored queries

none

Annex G
(informative)

Design rationale

The CSW catalogue profile is intended to provide a flexible, general-purpose catalogue service that can be adapted to meet the needs of a wide range of communities of practice. Key requirements:

- is broadly useful within the geospatial domain;
- provides powerful search capabilities, including spatial searches;
- can be tailored to meet the needs of specialized applications

Reference OWS-1.2 reqs?

Bibliography

DCMI, *DCMI Metadata Terms,* DCMI Recommendation (2003-11-19). Available [online]: <http://dublincore.org/documents/dcmi-terms/>.

ISO/IEC 8601:2000, *Data elements and interchange formats – Information interchange – Representation of dates and times.*

ISO/IEC 11179-3:2003, *Information technology – Metdata registries (MDR) – Part 3: Registry metamodel and basic attributes.*

ISO/WD 19135, *Geographic information – Procedures for registration of geographical information items.*

OASIS SAML, *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1*, OASIS Standard (2 September 2003). Available [online]: <http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf>.

OASIS XACML, *eXtensible Access Control Markup Language (XACML) Version 1.1*, OASIS Committee Specification (7 August 2003). Available [online]: <http://www.oasis-open.org/committees/xacml/repository/cs-xacml-specification-1.1.pdf>.

OGC 99-049, *OpenGIS Simple Features Specification for SQL*, Revision 1.1 (5 May 1999). Available [online]: <http://www.opengis.org/docs/99-049.pdf>.

OGC 02-023r4, *OpenGIS Geography Markup Language (GML) Implementation Specification*, Version 3.0 (29 January 2003). Available [online]: <http://www.opengis.org/docs/02-023r4.pdf>.

OMG 03-03-01, *Unified Modeling Language Specification*, Version 1.5. Available [online]: <http://www.omg.org/docs/formal/03-03-01.pdf>.

OWL-S, *OWL-S: Semantic Markup for Web Services,* version 1.0 (2003-11). Available [online]: <http://www.daml.org/services/owl-s/1.0/owl-s.html>.

RFC 2246, *The TLS protocol Version 1.0*, IETF Proposed Standard (January 1999). Available [online]: <http://www.ietf.org/rfc/rfc2246.txt>.

RFC 2387, *The MIME Multipart/Related Content-type*, IETF Proposed Standard (August 1998). Available [online]: <http://www.ietf.org/rfc/rfc2387.txt>.

RFC 2388, *Returning Values from Forms: multipart/form-data*, IETF Proposed Standard (August 1998). Available [online]: <http://www.ietf.org/rfc/rfc2388.txt>.

RFC 2392, *Content-ID and Message-ID Uniform Resource Locators*, IETF Proposed Standard (August 1998). Available [online]: <http://www.ietf.org/rfc/rfc2392.txt>.

RFC 2396, *Uniform Resource Identifiers (URI): Generic Syntax*, IETF Draft Standard (August 1998). Available [online]: <http://www.ietf.org/rfc/rfc2396.txt>.

RFC 2617, *HTTP Authentication: Basic and Digest Access Authentication*, IETF Draft Standard (June 1999). Available [online]: <http://www.ietf.org/rfc/rfc2617.txt>.

W3C SOAP-1, *SOAP Version 1.2 Part 1: Messaging Framework*, W3C Recommendation (24 June 2003). Available [online]: <http://www.w3.org/TR/SOAP/>.

W3C SOAP-2, *SOAP Version 1.2 Part 2: Adjuncts*, W3C Recommendation (24 June 2003). Available [online]: <http://www.w3.org/TR/soap12-part2/>.

W3C XPath1, *XML Path Language (XPath) Version 1.0*, W3C Recommendation (16 November 1999). Available [online]: <http://www.w3.org/TR/xpath>.

W3C XPointer, *XPointer Framework*, W3C Recommendation (25 March 2003). Available [online]: < http://www.w3.org/TR/xptr-framework/>.