Open GIS Consortium Inc.

Date: 2004-06-17

Reference number of this OpenGIS® document: OGC 04-016r3

Version: 0.3.0

Category: OpenGIS® Implementation Specification

Editor: Arliss Whiteside

OWS Common Implementation Specification

Copyright notice

Warning

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

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

Document type:	OpenGIS® Recommendation Paper
Document subtype:	Normatively referenced by other Implementation Specifications
Document stage:	Accepted Recommendation Paper
Document language:	English

File name: 04-016r3 OWS Common.doc

Contents

Page

Figures

Tables

i. Preface

This document specifies many of the aspects that are, or should be, common to all or multiple OGC Web Service (OWS) interface Implementation Specifications. These common aspects are primarily some of the parameters and data structures used in operation requests and responses. Of course, each such Implementation Specification must specify the additional aspects of that interface, including specifying all additional parameters and data structures needed in all operation requests and responses.

This document is currently an OGC Recommendation Paper, and is expected to become an Implementation Specification with the first Implementation Specification that normatively references it. That Implementation Specification is currently expected to be the OpenGIS® Catalog Services Specification version 2.0.

One expected use of this document is as a normative reference from all future versions of OWS interface Implementation Specifications. Rather than continuing to repeat this material in each such Implementation Specification, usually with small changes, each specification should normatively reference each relevant part of this document. Such normative references can take the form of stating “This TBD shall include TBD as specified in Subclause TBD of OGC document TBD.” Such normative references are expected to:

a) Reduce the work needed to edit and read each such Implementation Specification

b) Reduce the length of each such Implementation Specification

c) Increase interoperability among such Implementation Specifications by increasing commonality and discouraging non-essential differences

d) Provide useful guidance to writers of new and revised Implementation Specifications

Each existing OGC-approved and draft OWS interface Implementation Specification should consider this document to be a formal change request to modify that specification in its next revision to agree with all the relevant material specified herein. Each such specification is also requested to normatively reference each relevant part of this document (instead of repeating the same material).

Most of the current contents of this document have been agreed on by an ad-hoc OGC working group on OWS harmonization, initiated at the June 2003 OGC meeting. The current editors of the various OWS interface Implementation Specifications have participated in this OWS harmonization group. For most of the open issues not yet decided by that working group, this document contains an EDITOR’S NOTE that mentions the currently open issue(s).

This revision of this document contains small improvements in Subclauses 7.4.6, 7.4.7, 9.2.3, and 11.3, plus in Clause iii, Annex A, and the attached file owsGetCapabilities.xsd. All the non-formatting changes are tracked in Microsoft Word. This document is a draft proposed Implementation Specification. The previous version of this document was adopted as Recommendation Paper [OGC 03-088r6], and will be superseded by this Implementation Specification. The first version of this document was released as an OGC Discussion Paper 03-088r1, and was superseded by the Recommendation Paper.

Suggested additions, improvements, and comments on this specification are welcome and encouraged. Such suggestions may be submitted to the editor by email message. Extensive and/or multiple changes can be suggested by making changes in an edited copy of this document. If you choose to submit suggested changes by editing this document, please make your suggested changes with change tracking on.

ii. Subclauses ready for normative referencing

The subclauses in this version of this document that are believed to be sufficiently mature to be normatively referenced from other OGC Implementation Specifications are Clauses:

3	Normative references
4	Terms and definitions
5	Conventions
7	GetCapabilities operation
8	Exception reports
9	All operations except GetCapabilities, minimum abilities
10	Other operation parameters
11	Operation request and response encoding
A	XML schemas

iii. Document contributor contact points

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

Person	Company	Address	Phone	Email
Arliss Whiteside	BAE SYSTEMS Mission Solutions	10920 Technology Dr. San Diego, CA 92127-1608 USA	+1 858-592-1608	Arliss.Whiteside@baesystems.com
Jeff de La Beaujardière	NASA Goddard Space Flight Center	Code 933 Greenbelt MD 20771 USA	+1 301-286-1569	delabeau@iniki.gsfc.nasa.gov
Peter Vretanos	CubeWerx, Inc.	5 Wexford Blvd. Toronto, Ontario M1R 1K9 Canada	+1 416-701-1985	pvretano@cubewerx.com
John Evans	NASA / GST, Inc.		+1 301-286-0803	jdevans@gst.com
Keith Pomakis	CubeWerx, Inc.	200 rue Montcalm, Gatineau, Québec J8Y 3B5 Canada	+1 819-771-8303 x204	pomakis@cubewerx.com
Jeff Simon	General Dynamics - AIS		+1 703-668-3696	jeff.simon@gd-ais.com
Jerome Sonnet	IONIC Software S.A.	Rue de Wallonie, 18. B-4460 GRACE-HOLLOGNE - Belgium	+32.4.3640364	jerome.sonnet@ionicsoft.com
George Percivall	NASA / GST, Inc.	NASA Goddard Space Flight Center, Code 101.4	1+301-286-4073	percivall@gsfc.nasa.gov

iv. Revision history

Date	Release	Editor	Primary clauses modified	Description
2003-10-06	0.0.0	Arliss Whiteside	All	Initial version
2003-10-16	0.1.0	Arliss Whiteside	7.2.2.3, 7.4.3, 8.1	First approved discussion paper, small additions based on discussions in Architecture WG
2003-11-26	0.1.1	Arliss Whiteside	7.2.2, 7.2.3, 7.2.5, 8.1, 8.3, 11, A.2, A.4, B, C	Large additions to 7.2.5; significant editing of 7.4; addition of 11, B, and C; editing of most other parts
2003-12-22	0.1.2	Arliss Whiteside	6-11, A-C	Document reorganized, edited most parts
2004-01-05	0.1.3	Arliss Whiteside	5.1, 7, A-C	Document edited to correct errors and make clearer
2004-01-11	0.1.4	Arliss Whiteside	7.4, A-C	Modify two sections of service metadata
2004-01-15	0.2.0	Arliss Whiteside	7.4.2, 7.4.5, cover, i	Corrected Tables 7 and 13, edited to reflect approval as a Recommendation Paper
2004-03-05	0.2.1	Arliss Whiteside	ii, 10, 11, B, C.12 to C.14	Added specifications of bounding boxes and CRS references, added more information on encoding, miscellaneous editing
2004-03-29	0.2.2	Arliss Whiteside	3, 4.1, 7.3.3, 7.3.5, 7.4.2, 7.4.6, 7.4.7, 11.3, A, C.11	A few small changes plus many wording improvements
2004-02-12	0.2.3	Arliss Whiteside	iii, 7.4.6, 7.4.7, 9.2.3, 11.3, A, C.6	Various small improvements, changed owsCommon/xsd to owsGetCapabilities.xsd

v. Changes to the OpenGIS® Abstract Specification

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

vi. Future work

This document should be extended to include other aspects that should be common among multiple OWS Implementation Specifications, such as:

a) Improve organization of service metadata documents, such as to better match WSDL and UDDI

b) More of the contents of service metadata documents, such as for:

  1) Contents section

  2) Query language metadata

c) More parameters used in operation requests and responses, such as for:
    1) Output format
    2) Interpolation method
    3) Units
d) More common operations, such as for:
    1) Transaction
    2) Get(data)
    3) Describe(data)
e) More guidance for editing OWS Implementation Specifications
f) Expansion to handle chained services
g) Better accommodate use of various languages, when applicable in most operation requests and responses

Foreword

This OGC Implementation Specification supersedes and replaces OGC Recommendation Paper 03-088r6. This version has been expanded and edited.

This document includes three annexes; Annex A in normative, and Annexes B and C are informative.

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

Introduction

This document specifies many of the aspects that are common to all or multiple OWS interface Implementation Specifications. Those specifications currently include the Web Map Service (WMS), Web Feature Service (WFS), and Web Coverage Service (WCS). These common aspects include: operation request and response contents; parameters included in operation requests and responses; and encoding of operation requests and responses.

OWS Common Implementation Specification

1 Scope

This document specifies many of the aspects that are, or should be, common to all or multiple OWS interface Implementation Specifications. The common Implementation Specification aspects specified by this document currently include:

a) Operation request and response contents, most partial

b) Parameters included in operation requests and responses

c) XML and KVP encoding of operation requests and responses

One use of this document is as a normative reference from future versions of OWS interface Implementation Specifications. Those specifications currently include the Web Map Service (WMS), Web Feature Service (WFS), and Web Coverage Service (WCS). Rather than continuing to repeat this material in each such Implementation Specification, each specification should normatively reference each relevant part of this document.

2 Conformance

Conformance with this specification shall be checked using all the relevant tests specified in each separate specification that normatively references this specification, and specifically references the applicable parts of this specification.

3 Normative references

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

IETF RFC 2045 (November 1996), *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, Freed, N. and Borenstein N., eds., <http://www.ietf.org/rfc/rfc2045.txt>

IETF RFC 2141 (May 1997), *URN Syntax*, R. Moats
<http://www.ietf.org/rfc/rfc2141.txt>

IETF RFC 2396 (August 1998), *Uniform Resource Identifiers (URI): Generic Syntax*, Berners-Lee, T., Fielding, N., and Masinter, L., eds., <http://www.ietf.org/rfc/rfc2396.txt>

IETF RFC 2616 (June 1999), *Hypertext Transfer Protocol – HTTP/1.1*, Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T., eds., <http://www.ietf.org/rfc/rfc2616.txt>

IANA, Internet Assigned Numbers Authority, MIME Media Types, available at <http://www.iana.org/assignments/media-types/>

ISO 4217:2001, *Codes for the representation of currencies and funds*

ISO 8601:2000(E), *Data elements and interchange formats - Information interchange - Representation of dates and times*.

ISO 19115:2003, *Geographic information — Metadata*

ISO 19119:TBD, *Geographic information — Services*

OGC Topic 2, OGC 03-073r4, The OpenGIS Abstract Specification, Topic 2: Spatial Referencing by Coordinates, October 2003

OGC 03-105r1, OpenGIS Geography Markup Language (GML) Implementation Specification, Version 3.1, February 2004

OGC 03-107r1, OpenGIS Geography Markup Language (GML) Implementation Specification Schemas, Version 3.1, February 2004

W3C Recommendation January 1999, *Namespaces In XML*, http://www.w3.org/TR/2000/REC-xml-names.

W3C Recommendation 6 October 2000, *Extensible Markup Language (XML) 1.0* (Second Edition), http://www.w3.org/TR/REC-xml

W3C Recommendation 2 May 2001: *XML Schema Part 0: Primer*, http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/

W3C Recommendation 2 May 2001: *XML Schema Part 1: Structures*, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

W3C Recommendation 2 May 2001: *XML Schema Part 2: Datatypes*, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

In addition to this document, this specification includes several normative XML Schema files. These are posted online at the URL http://schemas.opengis.net/ows/. These XML Schema files are also bundled with the present document. In the event of a discrepancy between the bundled and online versions of the XML Schema files, the online files shall be considered authoritative.

4 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

4.1
bounding box

portion of a coordinate space that lies between a lower bound and an upper bound in each dimension of a coordinate reference system

NOTE A bounding box can be used to express spatial-temporal query constraints, or to describe the (approximate) location and extent of geospatial data. A bounding box is often called the "minimum bounding rectangle" of a geospatial data item when its lower and upper bounds in each dimension are those of the data item.

EXAMPLES Rectangle in two spatial dimensions, rectangular solid in three spatial dimensions

4.2
capabilities XML

service metadata encoded in XML

4.3
client

software component that can invoke an **operation** from a **server**

4.4
geographic information

information concerning phenomena implicitly or explicitly associated with a location relative to the Earth [ISO 19128 draft]

4.5
interface

named set of operations that characterize the behaviour of an entity [ISO 19119]

4.6
operation

specification of a transformation or query that an object may be called to execute [ISO 19119]

4.7
parameter

variable whose name and value are included in an operation **request** or **response**

4.8
request

invocation of an **operation** by a **client**

4.9
response
result of an **operation,** returned from a **server** to a **client**

4.10
server
service instance
a particular instance of a **service** [ISO 19119 edited]

4.11
service
distinct part of the functionality that is provided by an entity through interfaces [ISO 19119]

capability which a service provider entity makes available to a service user entity at the interface between those entities [ISO 19104 terms repository]

4.12
service metadata
metadata describing the **operations** and **geographic information** available at a **server** [ISO 19128 draft]

4.13
version
version of an Implementation Specification (document) and XML Schemas to which the requested operation conforms

NOTE An OWS Implementation Specification version may specify XML Schemas against which an XML encoded operation request or response must conform and should be validated.

5 Conventions

5.1 Symbols (and abbreviated terms)

CRS	Coordinate Reference System
DCP	Distributed Computing Platform
EPSG	European Petroleum Survey Group
GIS	Geographic Information System
GML	Geography Markup Language
HTTP	Hypertext Transfer Protocol
ISO	International Organization for Standardization
KVP	Keyword Value Pair
MIME	Multipurpose Internet Mail Extensions

OGC	Open GIS Consortium
OWS	OGC Web Service, or Open Web Service
TBD	To Be Determined
TBR	To Be Reviewed
UML	Unified Modeling Language
URI	Universal Resource Identifier
URL	Uniform Resource Locator
URN	Universal Resource Name
WCS	Web Coverage Service
WFS	Web Feature Service
WMS	Web Map Service
XML	Extensible Markup Language
1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional
4D	Four Dimensional

5.2 UML notation

Some of the diagrams in this document are presented using the Unified Modeling Language (UML) static structure diagram. The UML notations used in this document are described in the diagram below.

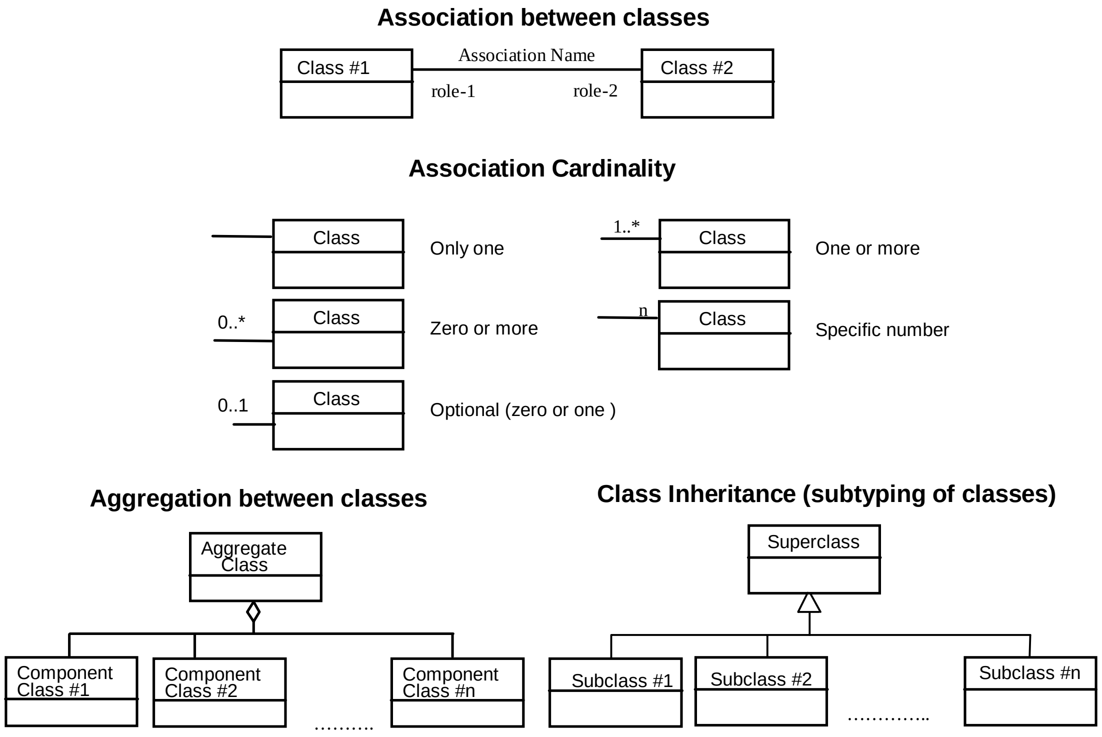

Figure 1 — UML notations

In these UML class diagrams, the class boxes with a light background are the primary classes being shown in this diagram, often the classes from one UML package. The class boxes with a gray background are other classes used by these primary classes, usually classes from other packages.

In this diagram, the following stereotypes of UML classes are used:

a) <<DataType>> A descriptor of a set of values that lack identity (independent existence and the possibility of side effects). A DataType is a class with no operations, whose primary purpose is to hold the information.

b) <<Enumeration>> A data type whose instances form a list of alternative literal values. Enumeration means a short list of well-understood potential values within a class.

c) <<CodeList>> A flexible enumeration for expressing a long list of potential alternative values. If the list alternatives are completely known, an enumeration shall be used; if the only likely alternatives are known, a code list shall be used.

d) <<Interface>> A definition of a set of operations that is supported by objects having this interface. An Interface class cannot contain any attributes.

e) <<Type>> A stereotyped class used for specification of a domain of instances (objects), together with the operations applicable to the objects. A Type class may have attributes and associations.

f) <<Union>> A list of alternate attributes where only one of those attributes can be present at any time.

NOTE All the stereotypes listed above are adapted from Subclauses 6.8.2 and D.8.3 of ISO 19103.

In this document, the following standard data types are used:

a) CharacterString – A sequence of characters
b) Boolean – A value specifying TRUE or FALSE
c) URI – An identifier of a resource that provides more information
d) URL – An identifier of an on-line resource that can be electronically accessed
e) Integer – An integer number
f) Double – A double precision floating point number

5.3 Document terms and definitions

The following specification terms and definitions are used in this document:

a) shall – verb form used to indicate a requirement to be strictly followed to conform to this specification, from which no deviation is permitted
b) should – verb form used to indicate desirable ability or use, without mentioning or excluding other possibilities
c) may – verb form used to indicate an action permissible within the limits of this specification
d) can – verb form used for statements of possibility
e) informative – a part of a document that is provided for explanation, but is not required
f) normative – a part of a standards document that is required
g) annex – an auxiliary part of a document, called an “appendix” in United States English
h) clause – a major part of a document, called a “section” or “paragraph” in United States English
i) subclause – a secondary part of a clause or annex, called a “subsection” in United States English

6 Document overview

This document is organized into clauses that discuss the:

a) GetCapabilities operation that is provided by all OWSs, currently including the parameters:
   1) “service”
   2) “request”
   3) “AcceptVersions”
   4) “Sections”
   5) “updateSequence”
   6) “AcceptFormats”

b) Exception report responses to all operation requests of all OWSs, including the parameters:
   1) “ExceptionText”
   2) “exceptionCode”
   3) “locator”
   4) “version”
   5) “language”

c) All operations except GetCapabilities, minimum parameters including:
   1) “service”
   2) “request”
   3) “version”

d) Other parameters used by multiple OWSs and operations, including the parameters:
   1) “BoundingBox”
   2) “WGS84BoundingBox”
   3) “LowerCorner”
   4) “UpperCorner”
   5) “dimensions”
   6) “crs”
   7) “CRS”

e) Encoding of OWS operation requests and responses

The annexes to this document provide related informative information on:

a) More complete XML Schemas, ready to use (normative)

b) UML model of aspects specified herein (informative)

c) Reasons for including various parameters (informative)

NOTE The following clauses and annexes are written to be relatively independent of one another. They can thus be read in any order, depending on reader knowledge and interests. For example, Clause 11 “Operation request and response encoding” contains detailed information supporting Clauses 7 through 10, and can be read first or last.

7 GetCapabilities operation

7.1 Introduction

This clause partially specifies the GetCapabilities operation provided by each OWS. The mandatory GetCapabilities operation allows any client to retrieve metadata about the services available from any server that implements an OWS interface Implementation Specification. The normal response to the GetCapabilities operation is a service metadata document that is returned to the requesting client. This service metadata document primarily contains metadata about the specific server abilities (such as about the specific data and formats available from that server). This service metadata also makes an OWS server partially self-describing, supporting late binding of clients.

NOTE A specific OWS Implementation Specification or implementation can provide additional operation(s) returning service metadata for a server. Such operations can return service metadata using different data structures and/or formats, such as WSDL or ebRIM. When such operation(s) have been sufficiently specified and shown more useful, the OGC may decide to require those operation(s) instead of the current GetCapabilities operation.

7.2 GetCapabilities request

7.2.1 GetCapabilities request parameters

A request to perform the GetCapabilities operation shall include the parameters listed and defined in Table 1. This table also specifies the UML model data type, source of values, and optionality of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request.

Table 1 — Parameters in GetCapabilities operation request

Name [a]	Definition	Data type and value	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is OWS type abbreviation (e.g., “WMS”, “WFS”)	One (mandatory)
request	Operation name	Character String type, not empty Value is operation name (e.g., “GetCapabilities”)	One (mandatory)
AcceptVersions	Prioritized sequence of one or more specification versions accepted by client, with preferred versions listed first	Sequence of Character String type, each not empty Value is list of x.y.z “version” values	Zero or one (optional) When omitted, return latest supported version (see Subclause 7.3.2)
Sections	Unordered list of zero or more names of requested sections in complete service metadata document [b]	Sequence of Character String type, each not empty Value is list of section names Allowed section names are specified by each Implementation Specification	Zero or one (optional) When omitted or not supported by server, return complete service metadata document
updateSequence	Service metadata document version, value is “increased” whenever any change is made in complete service metadata document	Character String type, not empty Values are selected by each server, and are always opaque to clients	Zero or one (optional) When omitted or not supported by server, return latest service metadata document
AcceptFormats	Prioritized sequence of zero or more response formats desired by client, with preferred formats listed first	Sequence of Character String type, each not empty Value is list of format identifiers Identifiers are MIME types of formats useful for service metadata documents	Zero or one (optional) When omitted or not supported by server, return service metadata document using MIME type "text/xml"

[a] Although some values listed in the “Name” column appear to contain spaces, they shall not contain spaces.

[b] The “Sections” parameter specifies which XML elements within a service metadata document shall be returned, within an (usually abbreviated) “Capabilities” element. The allowed section name values shall be specified by each Implementation Specification, as specified in Subclause 7.3.3.

NOTE 1 The name capitalization rules being used here are specified in Subclause 11.6.2.

NOTE 2 The data type of many parameters is specified as “Character String type, not empty”. In the XML Schemas specified herein, these parameters are encoded with the xsd:string type, which does NOT require that these strings not be empty.

7.2.2 GetCapabilities request KVP encoding

An Implementation Specification fragment for KVP encoding of the GetCapabilities operation request, with example values appropriate for WCS 1.0.0, is shown in Table 2.

Table 2 — GetCapabilities operation request URL parameters

Name and example [a]	Optionality and use	Definition and format
service=WCS	Mandatory	Service type identifier text
request=GetCapabilities	Mandatory	Operation name text
AcceptVersions=1.0.0,0.8.3	Optional When omitted, return latest supported version (see Subclause 7.3.2)	Comma-separated prioritized sequence of one or more specification versions accepted by client, with preferred versions listed first
Sections=Contents	Optional When omitted or not supported by server, return complete service metadata document	Comma-separated unordered list of zero or more names of sections of service metadata document to be returned in service metadata document
updateSequence=XXX (where XXX is character string previously provided by server)	Optional When omitted or not supported by server, return latest service metadata document version	Service metadata document version, value is “increased” whenever any change is made in complete service metadata document
AcceptFormats= text/xml	Optional When omitted or not supported by server, return service metadata document using MIME type "text/xml"	Comma-separated prioritized sequence of zero or more response formats desired by client, with preferred formats listed first

[a] All parameter names are here listed using mostly lower case letters. However, any parameter name capitalization shall be allowed in KVP encoding, see Subclause 11.5.2.

In a specific OWS Implementation Specification, this table shall be supported by a specification of the section names allowed in the Sections parameter, with the meaning of each value for that specific OWS. These section names and meanings shall be based on Subclause 7.4.2.

A corresponding example of a GetCapabilities request message encoded using KVP is:

```
http://hostname:port/path?SERVICE=WCS&REQUEST=GetCapabilities&ACCEPTVER
   SIONS=1.0.0,0.8.3&SECTIONS=Contents&UPDATESEQUENCE=XYZ123&
   ACCEPTFORMATS=text/xml
```

This example includes all six possible parameters, but only the “service” and “request” parameters are required.

7.2.3 GetCapabilities request XML encoding

The XML Schema fragment for encoding a generic GetCapabilities operation request is:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.opengis.net/ows"
xmlns:ows="http://www.opengis.net/ows"
```

```
xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
version="0.1.3" xml:lang="en">
   <annotation>
      <appinfo>owsGetCapabilitiesRequest.xsd 2004/1/02</appinfo>
      <documentation>
         <description>This XML Schema encodes the GetCapabilities
operation request. This XML Schema must be edited by each OWS, to
specify a specific value for the "service" attribute. Primary editor:
Arliss Whiteside. </description>
         <copyright>Copyright (c) 2003 OpenGIS, All Rights Reserved.
</copyright>
      </documentation>
   </annotation>
   <!-- ==============================================================
      elements and types
   ============================================================== -->
   <element name="GetCapabilities" type="ows:GetCapabilitiesType"/>
   <!-- =========================================================== -->
   <complexType name="GetCapabilitiesType">
      <annotation>
         <documentation>XML encoded GetCapabilities operation request.
This operation allows clients to retrieve service metadata about a
specific service instance. In this XML encoding, no "request" parameter
is included, since the element name specifies the specific operation.
</documentation>
      </annotation>
      <sequence>
         <element name="AcceptVersions" type="ows:AcceptVersionsType"
minOccurs="0">
            <annotation>
               <documentation>When omitted, server shall return latest
supported version. </documentation>
            </annotation>
         </element>
         <element name="Sections" type="ows:SectionsType"
minOccurs="0">
            <annotation>
               <documentation>When omitted or not supported by server,
server shall return complete service metadata (Capabilities) document.
</documentation>
            </annotation>
         </element>
         <element name="AcceptFormats" type="ows:AcceptFormatsType"
minOccurs="0">
            <annotation>
               <documentation>When omitted or not supported by server,
server shall return service metadata document using the MIME type
"text/xml". </documentation>
            </annotation>
         </element>
      </sequence>
      <attribute name="service" type="ows:ServiceType" use="required"/>
      <attribute name="updateSequence" type="ows:UpdateSequenceType"
use="optional">
         <annotation>
```

```
            <documentation>When omitted or not supported by server, 
server shall return latest complete service metadata document. 
</documentation>
         </annotation>
      </attribute>
   </complexType>
   <!-- ========================================================== -->
   <simpleType name="ServiceType">
      <annotation>
         <documentation>Service type identifier, where the string value 
is the OWS type abbreviation, such as "WMS" or "WFS". </documentation>
      </annotation>
      <restriction base="string"/>
   </simpleType>
   <!-- ========================================================= -->
   <complexType name="AcceptVersionsType">
      <annotation>
         <documentation>Prioritized sequence of one or more 
specification versions accepted by client, with preferred versions 
listed first. See Version negotiation subclause for more information. 
</documentation>
      </annotation>
      <sequence>
         <element name="Version" type="ows:VersionType" 
maxOccurs="unbounded"/>
      </sequence>
   </complexType>
   <!-- ========================================================= -->
   <simpleType name="VersionType">
      <annotation>
         <documentation>Specification version for a WCS operation. The 
string value shall contain one x.y.z "version" value (e.g., "2.1.3"). A 
version number shall contain three non-negative integers separated by 
decimal points, in the form "x.y.z". The integers y and z shall not 
exceed 99. Each version shall be for the Implementation Specification 
(document) and the associated XML Schemas to which requested operations 
will conform. An Implementation Specification version normally 
specifies XML Schemas against which an XML encoded operation response 
must conform and should be validated. See Version negotiation subclause 
for more information. </documentation>
      </annotation>
      <restriction base="string"/>
   </simpleType>
   <!-- ========================================================== -->
   <complexType name="SectionsType">
      <annotation>
         <documentation>Unordered list of zero or more names of 
requested sections in complete service metadata document. Each Section 
value shall contain an allowed section name as specified by each OWS 
specification. See Sections parameter subclause for more information. 
</documentation>
      </annotation>
      <sequence>
         <element name="Section" type="string" minOccurs="0" 
maxOccurs="unbounded"/>
      </sequence>
   </complexType>
   <!-- ============================================================ -->
   <simpleType name="UpdateSequenceType">
      <annotation>
         <documentation>Service metadata document version, having
values that are "increased" whenever any change is made in service
metadata document. Values are selected by each server, and are always
opaque to clients. See updateSequence parameter use subclause for more
information. </documentation>
      </annotation>
      <restriction base="string"/>
   </simpleType>
   <!-- ============================================================ -->
   <complexType name="AcceptFormatsType">
      <annotation>
         <documentation>Prioritized sequence of zero or more
GetCapabilities operation response formats desired by client, with
preferred formats listed first. Each response format shall be
identified by its MIME type. See AcceptFormats parameter use subclause
for more information. </documentation>
      </annotation>
      <sequence>
         <element name="OutputFormat" type="ows:FormatType"
minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
   </complexType>
   <!-- ============================================================ -->
   <simpleType name="FormatType">
      <annotation>
         <documentation>Data transfer format identifier, identified by
its MIME type. </documentation>
      </annotation>
      <restriction base="string"/>
   </simpleType>
</schema>
```

This XML Schema fragment contains documentation of the meaning of each element, attribute, and type, and this documentation shall be considered normative as specified in Subclause 11.6.3.

A corresponding example of a GetCapabilities request message encoded in XML is:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetCapabilities xmlns="http://www.opengis.net/ows"
xmlns:ows="http://www.opengis.net/ows"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/ows
owsGetCapabilitiesRequest.xsd" service="WCS" updateSequence="XYZ123">
   <!-- Maximum example for WCS. Primary editor: Arliss Whiteside. Last
updated 2003/12/20. -->
   <AcceptVersions>
      <Version>1.0.0</Version>
      <Version>0.8.3</Version>
   </AcceptVersions>
   <Sections>
      <Section>Contents</Section>
   </Sections>
   <AcceptFormats>
      <OutputFormat>text/xml</OutputFormat>
   </AcceptFormats>
</GetCapabilities>
```

This example includes all of the possible XML attributes and elements, but only the "service" attribute is required, within the required GetCapabilities root element.

7.3 Parameter discussions

7.3.1 Version parameter

Each OWS Implementation Specification revision shall specify a version number, which enables interacting clients and servers to agree on which version of the specification they are conforming to. A version number shall contain three non-negative integers separated by decimal points, in the form "x.y.z". The integers y and z shall not exceed 99.

Through the evolution of specifications, each service will have a number of versions defined for it, each with a different version number. Each OWS shall have its own sequence of version numbers; the version numbers of different services are independent and therefore may overlap. When the protocol version number changes, it shall increase monotonically, with the first integer being the most significant. There may be gaps in the numerical sequence, and some numbers may denote draft versions. Servers and their clients need not support all defined versions, but are encouraged to support multiple versions.

7.3.2 Version negotiation

Version negotiation is performed using the optional AcceptVersions parameter in the GetCapabilities operation request. Although optional, client software should always include this parameter, to simplify version negotiation. The value of this parameter is a sequence of protocol version numbers that the client supports, in order of client preference.

The server, upon receiving a GetCapabilities request, shall scan through this list and find the first version number that it supports. It shall then return a service metadata document conforming to that version of the specification, and containing that value of the "version" parameter. If the list does not contain any version numbers that the server supports, the server shall return an Exception with exceptionCode="VersionNegotiationFailed".

To ensure backward compatibility, clients shall also be prepared to accept an unknown response and treat this situation as an indication that version negotiation has failed. Furthermore, if a server receives a GetCapabilities request without the AcceptVersions parameter, it shall return a service metadata document that is compliant to the highest protocol version that the server supports. This makes it convenient for humans to make requests manually, and allows for forward compatibility with possible future incarnations of version negotiation.

This new version negotiation process is designed to be compatible with the old-style version negotiation that was defined in earlier versions of the various OWS specifications, as described in Subclause C.11.

7.3.3 Sections parameter

The Sections parameter value shall contain an unordered list of zero or more names, of the XML elements within a service metadata XML document that shall be returned. When one or more names are listed, those section(s) shall be included within a (usually abbreviated) service metadata document returned. If no names are listed, the service metadata returned may not contain any of the sections that could be listed.

The allowed section name values shall be specified in each Implementation Specification, and shall include, but are not limited to, all the values specified in Table 3 that are relevant to the specific OWS. The values allowed shall include “All”.

Table 3 — Meanings of section name values

Section name	Meaning
ServiceIdentification	Return ServiceIdentification element in service metadata document
ServiceProvider	Return ServiceProvider metadata element in service metadata document
OperationsMetadata	Return OperationsMetadata element in service metadata document
Contents	Return Contents metadata element in service metadata document
All	Return complete service metadata document, containing all elements

NOTE 1 All of the section name values listed in Table 3 are expected to be common for all OWSs, but some can add additional sections.

Client implementation of the Sections parameter is optional. When any server receives a GetCapabilities operation request without this parameter, it shall return the complete service metadata document.

Server implementation of the Sections parameter is optional. When a server does not implement this Sections parameter, it shall ignore this parameter if present in a GetCapabilities operation request, and shall return the complete service metadata document.

NOTE 2 A referencing OGC Implementation Specification is expected to leave optional the implementation of the Sections parameter, by both servers and clients. This flexibility allows Implementation Specification profiles to make server implementation of this parameter either required or prohibited.

7.3.4 updateSequence parameter

The optional updateSequence parameter can be used for maintaining the consistency of a client cache of the contents of a service metadata document. The parameter value can be an integer, a timestamp in [ISO 8601:2000] format, or any other number or string. The server may include an updateSequence value in its service metadata document. If supported, the updateSequence value shall be increased by the server when any changes are made to the complete service metadata document (for example, when new coverages are added to the WCS service). The server is the sole judge of lexical ordering sequence. The client may include this parameter in its GetCapabilities request. The response of the server based on the presence and relative value of updateSequence in the client request and the server metadata shall be as specified in Table 4.

Table 4 — Use of updateSequence parameter

Operation request updateSequence value	Service metadata updateSequence value	Server response
None	Any	most recent service metadata document
Any	None	most recent service metadata document
Equal	Equal	service metadata document with only “version” and “updateSequence” parameters
Lower	Higher	most recent service metadata document
Higher	Lower	exception report with exceptionCode = InvalidUpdateSequence

7.3.5 AcceptFormats parameter

The optional AcceptFormats parameter can be used by a client to attempt to negotiate a GetCapabilities operation response format other than "text/xml". When included in an operation request, this parameter shall contain a list of the alternative MIME types that the client wants to be returned, listed in the client's preferred order. The MIME type "text/xml" is always an implicit last option, but can be explicitly included.

When a server implements the AcceptFormats parameter and receives a value for it, the server shall return the Capabilities document in the format of the first MIME type in this list that it is capable of returning. When not received or not implemented, the server shall return the Capabilities document in normal XML, using the MIME type "text/xml". All clients and servers shall implement the "text/xml" MIME type for the GetCapabilities operation. Since "text/xml" is always an implicit last option, the server always has an implemented MIME type to use to return a Capabilities document to the client.

Server and client implementation of this parameter is optional. A variety of alternative formats (with different MIME types) have been proposed for transfer of XML documents, but many have not yet been completely specified, and none has yet been widely accepted. Many of these alternative formats reduce the size of the transferred message, thus reducing the communication time and load.

This document does not now specify any alternative format, but the AcceptFormats parameter is included to provide flexibility to allow experimentation and allow other documents to identify allowed alternative format(s). A specific OWS Implementation Specification that expects to interoperably use this AcceptFormats parameter shall thus identify the alternative format(s) that can be used (or that shall be implemented by servers).

EXAMPLE 1 One possible alternative format is the ISO standard for binary encoding of MPEG-7 or “BiM” as specified in [ISO/IEC 15938-1], with MIME type "application/x-bix".

EXAMPLE 2 Another possible alternative format is “BXML” as specified in [OGC 03-002r8], with MIME type "application/x-bxml".

NOTE A non-XML format whose MIME type is well-defined might be used if a method is specified to convert a Capabilities XML document, as specified herein, into that alternative format.

7.4 GetCapabilities response

7.4.1 Exceptions

In the event that an OWS server encounters an error servicing a GetCapabilities operation request, it shall return an exception report message as specified in Clause 8. The allowed exception codes shall include those listed in Table 5, assuming the updateSequence parameter is implemented by the server.

Table 5 — Exception codes for GetCapabilities operation

exceptionCode value	Meaning of code	“locator” value
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
VersionNegotiationFailed	List of versions in “AcceptVersions” parameter value in GetCapabilities operation request did not include any version supported by this server	None, omit “locator” parameter
InvalidUpdateSequence	Value of (optional) updateSequence parameter in GetCapabilities operation request is greater than current value of service metadata updateSequence number	None, omit “locator” parameter
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit “locator” parameter

7.4.2 Service metadata document contents

A service metadata document shall be the normal response to a client from performing the GetCapabilities operation, and shall contain metadata appropriate to the specific server for the specific OWS. For a server with tightly coupled data that it serves or uses, this service metadata document shall include metadata about that data. That service metadata document shall be encoded in XML, and shall use XML Schemas to specify the correct document contents and organization.

NOTE The term “Capabilities XML” document was previously usually used for what is here called “service metadata” document. The term “service metadata” is now used because it is more descriptive and is compliant with OGC Abstract specification topic 12 (ISO 19119). This “service metadata” includes metadata for a specific server and for tightly coupled data that it serves.

Each service metadata document shall include, in addition to other data, the parameters named and defined in Table 6. This table also specifies the data type, source of values, and multiplicity of each listed parameter, plus the meaning to the client when each optional parameter is not included in the service metadata document.

Table 6 — Parameters included in service metadata document

Name	Definition	Data type and value	Multiplicity and use
version	Specification version for operation, in this case for GetCapabilities operation response	Character String type, not empty Value is specified by each Implementation Specification and Schemas version (see Subclause 7.3.1)	One (mandatory)
updateSequence	Service metadata document version, value is “increased” whenever any change is made in complete service metadata document	Character String type, not empty Values are selected by each server, and are always opaque to clients	Zero or one (optional) Omitted when parameter not supported by server

Each service metadata document shall include a set of document sections that correspond to the set of section names specified for that specific OWS, as specified in Subclause 7.3.3 and used in the Sections parameter specified in Subclause 7.2. The common set of section names and meanings shall be as specified in Table 7. Each specific OWS shall use these section names and meanings when relevant, and can specify additional sections when needed.

Table 7 — Section name values and contents

Section name	Contents
ServiceIdentification	Metadata about this specific server. The schema of this section shall be the same for all OWSs.
ServiceProvider	Metadata about the organization operating this server. The schema of this section shall be the same for all OWSs.
OperationsMetadata	Metadata about the operations specified by this service and implemented by this server, including the URLs for operation requests. The basic contents and organization of this section shall be the same for all OWSs, but individual services can add elements and/or change the optionality of optional elements.
Contents	Metadata about the data served by this server. The schema of this section is specific to each OWS type, as defined by that Implementation Specification. Whenever applicable, this section shall contain a set of dataset descriptions, which should each be based on the MD_DataIdentification class specified in ISO 19115 and used in ISO 19119.

The allowed section names with their meanings should be specified in an Implementation Specification using a table such as Table 7 above, or by referencing this subclause and table. Most of the section name values listed in Table 7 are expected to be common for all OWSs.

7.4.3 ServiceIdentification section contents

The ServiceIdentification section of a service metadata document contains basic metadata about this specific server. The contents and organization of this section shall be the same for all OWSs. The ServiceIdentification section shall include the parameters and parts listed in Table 8. This table also specifies the data type and multiplicity of each listed parameter, plus the meaning to the client of that multiplicity.

Table 8 — Parameters included in ServiceIdentification section

Name [a]	Definition	Data type	Multiplicity and use
ServiceType	A service type name from registry of services, normally used for machine-to-machine communication	Character string type, not empty	One (mandatory)
ServiceType Version	Version of this service type implemented by this server	Character string type, not empty	One or more (mandatory) One for each version implemented by server, unordered
Title	Title of this server, normally used for display to a human	Character string type, not empty	One (mandatory)
Abstract	Brief narrative description of this server, normally available for display to a human	Character string type, not empty	Zero or one (optional) Include when server chooses, recommended and usually included
Keywords	Unordered list of one or more commonly used or formalised word(s) or phrase(s) used to describe this server	See MD_Keywords class in ISO 19115	Zero or more (optional) One for each keyword authority used
Fees	Fees and terms for retrieving data from or otherwise using this server, including the monetary units as specified in ISO 4217	Character string type, not empty Reserved value NONE shall be used to mean no fees or terms	Zero or one (optional)) Include when server chooses, recommended and usually included
Access Constraints	Access constraints that should be observed to assure the protection of privacy or intellectual property, and any other restrictions on retrieving or using data from or otherwise using this server	Character string type, not empty Reserved value NONE shall be used to mean no constraints are imposed	Zero or more (optional) Include when server chooses, recommended and usually included

[a] Although some values listed in the “Name” column appear to contain spaces, they shall not contain spaces.

As indicated, the Keywords parameter listed in Table 8 shall have contents based on the corresponding class in ISO 19115: Metadata (and OGC Abstract Specification Topic 11). With the exception of the ServiceType, all parameters contain server-specific information (not general service information). More detailed information on the contents and uses of all listed parameters is provided in the owsServiceIdentification.xsd XML Schema in Subclause 7.4.7.

7.4.4 ServiceProvider section contents

The ServiceProvider section of a service metadata document contains metadata about the organization operating this server. The contents and organization of this section shall be the same for all OWSs. The ServiceProvider section shall include the parameters and parts listed in Table 9. This table also specifies the data type and multiplicity of each listed parameter, plus the meaning to the client of that multiplicity.

Table 9 — Parameters included in ServiceProvider section

Name	Definition	Data type	Multiplicity and use
ProviderName	Unique identifier for service provider organization	Character string type, not empty	One (mandatory)
ProviderSite	Reference to the most relevant web site of the service provider	See CI_OnlineResource class in ISO 19115	Zero or one (optional) Include when useful
ServiceContact	Information for contacting service provider	See CI_ResponsibleParty and subsidiary classes in ISO 19115 [a]	One (mandatory)

a The contents of the CI_ResponsibleParty class are modified to omit the optional organizationName attribute in CI_ContactInfo, since the ProviderName contains this information. The optional individualName element is made mandatory, since either the organizationName or individualName element is mandatory. Also, the mandatory “role” attribute in the CI_ResponsibleParty class is made optional, since no clear use of this information is known in the ServiceProvider section.

As indicated, the ProviderSite and ServiceContact subsections listed in Table 9 shall have contents based on the corresponding classes in ISO 19115: Metadata (and OGC Abstract Specification Topic 11). More detailed information on the contents and uses of all listed parts is provided in the owsServiceProvider.xsd XML Schema referenced in Subclause 7.4.6.

7.4.5 OperationsMetadata section contents

The OperationsMetadata section of a service metadata document contains metadata about the operations provided by this service and implemented by this server, including the URLs for operation requests. The basic contents and organization of this section shall be the same for all OWSs, but individual services can add elements and/or change the optionality of optional elements. The OperationsMetadata section shall include the subsections listed in Table 10. This table also specifies the required multiplicity of each subsection, and the meaning to the client of that multiplicity.

Table 10 — Subsections included in OperationsMetadata section

Name	Definition	Multiplicity and use
Operation	Metadata for one operation that this server interface implements	One or more (mandatory) One for each implemented operation
Parameter	Parameter valid domain that applies to one or more operations which this server implements [a]	Zero or more (optional) One for each such parameter with limited domain
Constraint	Constraint on valid domain of a non-parameter quantity that applies to this server	Zero or more (optional) One for each such quantity with limited domain
ExtendedCapabilities	Metadata about server and software additional abilities	Zero or one (optional) Included when server provides additional capabilities
[a] This parameter can be an input and/or output parameter of these operations.		

The possible contents of the ExtendedCapabilities subsection are not specified here. The Operation, Parameter, and Constraint subsections shall include the parts specified in Tables 11-15. These tables also specify the data type, source of values, and multiplicity of each listed parameter, plus the meaning to the client of that multiplicity. More detailed information on the contents and uses of these parts is provided in the owsOperationsMetadata.xsd XML Schema referenced in Subclause 7.4.6.

Table 11 — Parts of Operation subsection

Name	Definition	Contents	Multiplicity and use
name	Name of this operation (request) (for example, GetCapabilities)	Character string type, not empty	One (mandatory)
DCP	Information for a Distributed Computing Platform (DCP) supported for this operation	See Table 12	One or more (mandatory) One for each supported DCP for this operation request [a]
Parameter	Parameter valid domain that applies to this operation which this server implements [b]	See Table 15	Zero or more (optional) One for each such parameter
Metadata	Metadata about this operation and its implementation [c]	Metadata contents or reference to metadata	Zero or more (optional) One for each such metadata object

[a] At present, only the HTTP DCP is defined, so the Operation subsection only includes one DCP subsection.

[b] This parameter can be an input and/or output parameter of this operation. If one of these Parameter subsections has the same parameter "name" as a Parameter subsection in the OperationsMetadata subsection, this Parameter subsection shall override the other one for this operation.

[c] Each operation that uses some form or query of filtering should include metadata describing the query or filter languages and associated capabilities implemented by this server. The schema of this query languages metadata is (currently) specific to each OWS type, as defined by that Implementation Specification.

Table 12 — Parts of DCP subsection

Name	Definition	Contents	Multiplicity and use
HTTP	Connect point URLs for the HTTP Distributed Computing Platform (DCP)	See Table 13	One (mandatory) [a]

[a] At present, only the HTTP DCP is defined, so the DCP subsection always includes the HTTP subsection.

Table 13 — Parts of HTTP subsection

Name	Definition	Contents	Multiplicity and use
Get	Connect point URL prefix for HTTP "Get" request method for this operation request	See ISO 19115 CI_OnlineResource type	Zero or more [a] One for each supported URL
Post	Connect point URL and accepted format(s) for HTTP "Post" request method for this operation request	See Table 14	Zero or more [a] One for each supported URL

[a] Normally, one Get and/or one Post is included in this subsection. More than one Get and/or Post is allowed to support including alternative URLs for uses such as load balancing or backup.

Table 14 — Parts of Post subsection

Name	Definition	Contents	Multiplicity and use
URL	Connect point URL for this operation request	See ISO 19115 CI_Online-Resource type	One (mandatory)
InputFormat	Format of operation request that can be sent to this URL	Character string type, not empty Values are MIME types	Zero or more (optional) When omitted, only “text/xml” is supported

Table 15 — Parts of Parameter and Constraint subsections

Name	Definition	Data type	Multiplicity and use
name	Name or identifier of this parameter or other quantity	Character string type, not empty	One (mandatory)
Value	Valid value for this parameter or other quantity [a]	Character string type, not empty	One or more (mandatory) One for each such value
Metadata	Metadata about domain of this parameter or other quantity	Metadata contents or reference to metadata	Zero or more (optional) One for each such metadata object
a For those parameters that contain a list or sequence of values, these values shall be for individual values in the list.			

7.4.6 Capabilities document XML encoding

In a “Capabilities” or service metadata XML document, all sections shall be encoded as XML elements, using the names and capitalization shown in Table 7. The XML Schema fragment for a generic service metadata document is:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.opengis.net/ows"
xmlns:ows="http://www.opengis.net/ows"
xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
version="0.1.4" xml:lang="en">
   <annotation>
      <appinfo>owsGetCapabilitiesResponse.xsd 2004/1/11</appinfo>
      <documentation>
         <description>This XML Schema encodes the GetCapabilities
operation response, also known as the Capabilities XML document. This
XML Schema must be expanded or edited by each OWS, to specify specific
contents of the Contents element and perhaps of the OperationsMetadata
element. </description>
         <copyright>Copyright (c) 2004 OpenGIS, All Rights Reserved.
</copyright>
      </documentation>
   </annotation>
   <!-- ==============================================================
      includes and imports
   ============================================================== -->
   <include schemaLocation="owsServiceIdentification.xsd"/>
   <include schemaLocation="owsServiceProvider.xsd"/>
   <include schemaLocation="owsOperationsMetadata.xsd"/>
   <!-- ==============================================================
      elements and types
   ============================================================== -->
   <element name="Capabilites" type="ows:CapabilitesType"/>
   <!-- ============================================================ -->
   <complexType name="CapabilitesType">
      <annotation>
         <documentation>XML encoded GetCapabilities operation response.
This document provides clients with service metadata about a specific
service instance, usually including metadata about the tightly-coupled
data served. If the server does not implement the updateSequence
parameter, the server shall always return the complete Capabilities
document, without the updateSequence parameter. When the server
implements the updateSequence parameter and the GetCapabilities
operation request included the updateSequence parameter with the
current value, the server shall return this element with only the
"version" and "updateSequence" attributes. Otherwise, all optional
elements shall be included or not depending on the actual value of the
Contents parameter in the GetCapabilities operation request.
</documentation>
      </annotation>
      <sequence>
         <element ref="ows:ServiceIdentification" minOccurs="0"/>
         <element ref="ows:ServiceProvider" minOccurs="0"/>
         <element ref="ows:OperationsMetadata" minOccurs="0"/>
         <element ref="ows:Contents" minOccurs="0">
            <annotation>
               <documentation>This element shall be included whenever
this OWS operates on any tightly-coupled data, by any specified
operation (even when that data is also available to clients from
another service). </documentation>
            </annotation>
         </element>
      </sequence>
      <attribute name="version" type="ows:VersionType" use="required"/>
      <attribute name="updateSequence" type="ows:UpdateSequenceType"
use="optional"/>
   </complexType>
   <!-- ============================================================ -->
   <element name="Contents" abstract="true">
      <annotation>
         <documentation>Metadata about the data served by this server.
The XML Schema of this section is specific to each OWS type.
</documentation>
      </annotation>
   </element>
   <!-- ============================================================ -->
   <simpleType name="UpdateSequenceType">
      <annotation>
         <documentation>Service metadata document version, having
values that are "increased" whenever any change is made in service
metadata document. Values are selected by each server, and are always
opaque to clients. See updateSequence parameter use subclause for more
information. </documentation>
      </annotation>
      <restriction base="string"/>
   </simpleType>
</schema>
```

The above schema fragment uses three separate schemas, named owsServiceIdentification.xsd, owsServiceProvider.xsd, and owsOperationsMetadata.xsd, which specify the contents of the ServiceIdentification, ServiceProvider, and OperationsMetadata sections. Also, the above XML Schema does not specify the contents of the Contents section, which are different for each OWS (and often for each version thereof).

The XML Schema for the standard “ServiceIdentification” section of Capabilities XML documents shall be as attached in the owsServiceIdentification.xsd file. This XML Schema uses parts of an XML encoding of ISO 19115 metadata, as specified in the attached ows19115subset.xsd file.

The XML Schema for the standard “ServiceProvider” section of Capabilities XML documents shall be as attached in the owsServiceProvider.xsd file. This XML Schema also uses parts of an XML encoding of ISO 19115 metadata, as specified in the attached ows19115subset.xsd file.

The XML Schema for the standard “OperationsMetadata” section of Capabilities XML documents shall be as attached in the owsOperationsMetadata.xsd file.

All these XML Schemas contain documentation of the meaning of each element, attribute, and type, and this documentation shall be considered normative as specified in Subclause 11.6.3.

7.4.7 OperationsMetadata section standard contents

Each Implementation Specification that normatively references the OperationsMetadata section shall specify the mandatory values to be included for various XML elements and attributes in the OperationsMetadata section. In addition, each such Specification should specify the optional values to be included for various XML elements and attributes in that section. These specifications should be in the form of tables such as Table 16. In addition to being an example table, the OWS common item listed in Table 16 shall be included in all such tables. The “Attribute name” column uses dot-separator notation to specify parts of a parent item. The “Attribute value” column references an operation parameter, and the meaning of including that value is listed in the right column.

Table 16 — Required values of OperationsMetadata section attributes

Attribute name	Attribute value	Meaning of attribute value
Operation.name	GetCapabilities	The GetCapabilities operation is implemented by this server.

There are many optional values of “name” attributes and “value” elements in the OperationsMetadata section, primarily for recording the domain of various parameters and quantities. For example, the domain of the exceptionCode parameter could record all the codes implemented for each operation by that specific server. Similarly, each of the

GetCapabilities operation optional request parameters might have its domain recorded. For example, the domain of the Sections parameter could record all the sections implemented by that specific server.

7.4.8 Service metadata XML example

A corresponding partial example of a “Capabilities” XML document (or GetCapabilities response message) encoded in XML is:

```
<?xml version="1.0" encoding="UTF-8"?>
<Capabilites xmlns="http://www.opengis.net/ows"
xmlns:ows="http://www.opengis.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/ows owsCommon.xsd"
version="1.2.0" updateSequence="ABC123">
   <!-- Partial example for WMS. Primary editor: Arliss Whiteside. Last
updated 2004/01/09. -->
   <ServiceIdentification>
      <ServiceType>OGC:WMS</ServiceType>
      <ServiceTypeVersion>1.2.0</ServiceTypeVersion>
      <ServiceTypeVersion>1.1.1</ServiceTypeVersion>
      <Title>Acme Corp. Map Server</Title>
      <Abstract>
      Map Server maintained by Acme Corporation.
      Contact: webmaster@wmt.acme.com.
      High quality maps showing roadrunner nests and possible ambush
locations. </Abstract>
      <Keywords>
         <Keyword>bird</Keyword>
         <Keyword>roadrunner</Keyword>
         <Keyword>ambush</Keyword>
      </Keywords>
      <Fees>NONE</Fees>
      <AccessConstraints>NONE</AccessConstraints>
   </ServiceIdentification>
   <ServiceProvider>
      <ProviderName>Acme Corporation</ProviderName>
      <ProviderSite xlink:href="http://hostname/"/>
      <ServiceContact>
         <IndividualName>Jeff Smith, Server
Administrator</IndividualName>
         <PositionName>Computer Scientist</PositionName>
         <ContactInfo>
            <Phone>
               <Voice>+1 301 555-1212</Voice>
               <Facsimile>+1 301 555-1212</Facsimile>
            </Phone>
            <Address>
               <DeliveryPoint>NASA Goddard Space Flight
Center</DeliveryPoint>
               <City>Greenbelt</City>
               <AdministrativeArea>MD</AdministrativeArea>
               <PostalCode>20771</PostalCode>
               <Country>USA</Country>
```

```
<ElectronicMailAddress>user@host.com</ElectronicMailAddress>
         </Address>
      </ContactInfo>
   </ServiceContact>
</ServiceProvider>
<OperationsMetadata>
   <Operation name="GetCapabilities">
      <DCP>
         <HTTP>
            <Get xlink:href="http://hostname:port/path?"/>
         </HTTP>
      </DCP>
      <Parameter name="Format">
         <Value>text/xml</Value>
      </Parameter>
   </Operation>
   <Operation name="GetMap">
      <DCP>
         <HTTP>
            <Get xlink:href="http://hostname:port/path?"/>
            <Post xlink:href="http://hostname:port/path?">
               <InputFormat>text/xml</InputFormat>
            </Post>
         </HTTP>
      </DCP>
      <Parameter name="Format">
         <Value>image/gif</Value>
         <Value>image/png</Value>
         <Value>image/jpeg</Value>
      </Parameter>
      <Parameter name="ExceptionFormat">
         <Value>text/xml</Value>
         <Value>text/plain</Value>
         <Value>text/html</Value>
         <Value>application/vnd.ogc.se_inimage</Value>
      </Parameter>
   </Operation>
   <Operation name="GetFeatureInfo">
      <DCP>
         <HTTP>
            <Get xlink:href="http://hostname:port/path?"/>
         </HTTP>
      </DCP>
      <Parameter name="Format">
         <Value>text/xml</Value>
         <Value>text/plain</Value>
         <Value>text/html</Value>
      </Parameter>
   </Operation>
   <Parameter name="ExceptionFormat">
      <Value>text/xml</Value>
      <Value>text/plain</Value>
      <Value>text/html</Value>
   </Parameter>
   <Constraint name="MaximumLayerLevels">
      <Value>5</Value>
      </Constraint>
      <Constraint name="MaximumWidth">
         <Value>4000</Value>
      </Constraint>
      <Constraint name="MaximumHeight">
         <Value>4000</Value>
      </Constraint>
   </OperationsMetadata>
</Capabilites>
```

8 Exception reports

8.1 Introduction

Upon receiving an invalid operation request, each OWS shall respond to the client using an Exception Report message to describe to the client application and/or its human user the reason(s) that the request is invalid. Whenever a server detects an exception condition while responding to a valid operation request, and cannot produce a normal response to that operation, the server shall also respond to the client using an Exception Report. This clause specifies the Exception Report response to all operation requests for all OWSs.

8.2 Exception report contents

Each Exception Report shall be encoded in XML and shall contain one or more Exception elements, with each such element signalling detection of an independent error. Each Exception element shall contain the parameters specified in Table 17. This table also specifies the data type, source of values, and multiplicity of each listed parameter, plus the optionality of each listed parameter, and the meaning to a client when each optional parameter is not included.

Table 17 — Parameters in Exception element

Name	Definition	Data type and value	Multiplicity and use
Exception Text	Text describing the specific exception represented by the exceptionCode	Character String type, not empty Value is exception description as defined by individual servers	Zero or more (optional) [a] Omitted only when no more useful information available
exception Code	Code representing the type of this exception	Character String type, not empty Allowed values are specified by each Implementation Specification and server implementation	One (mandatory)
locator	Indicator of the location in the client's operation request where this exception was encountered	Character String type, not empty Contents defined for each allowed exceptionCode value for each operation [b]	Zero or one (optional) Omitted when no useful value available
a When included, multiple ExceptionText values shall provide hierarchical information about one detected error, with the most significant information listed first. b The contents and meaning of this parameter shall be defined for each allowed exceptionCode value. For some exceptionCode values, the meaning may be different for different operations. This locator should be included whenever meaningful information can be provided by the server.			

In addition to the Exception elements, an Exception Report shall also contain the parameters specified in Table 18.

Table 18 — Additional parameters in Exception Report

Name	Definition	Data type and value	Multiplicity and use
version	Specification version, in this case the version to which this Exception Report conforms	Character String type, not empty Value format is x.y.z, where x, y, and z are non-negative integers separated by decimal points (e.g., “2.1.3”) Value is specified by each Implementation Specification and Schemas version	One (mandatory)
language	Language used by all included exception text values	Character String type, not empty Values are language codes as specified by IETF RFC 1766	Zero or one (optional) Should be included

8.3 exceptionCode parameter values

Each Implementation Specification shall specify a set of standard allowed values for the exceptionCode parameter, as needed for each operation specified for that OWS. For each operation, the allowed standard exceptionCode values shall include all the relevant values specified in Table 19. The allowed standard exceptionCode values for each operation should be specified in a table such as Table 19. (The right column of Table 19 is described in the following subclause.)

Table 19 — Standard exception codes and meanings

exceptionCode value	Meaning of code	"locator" value
OperationNotSupported	Request is for an operation that is not supported by this server	Name of operation not supported
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value [a]	Name of parameter with invalid value
VersionNegotiationFailed	List of versions in "AcceptVersions" parameter value in GetCapabilities operation request did not include any version supported by this server	None, omit "locator" parameter
InvalidUpdateSequence	Value of (optional) updateSequence parameter in GetCapabilities operation request is greater than current value of service metadata updateSequence number	None, omit "locator" parameter
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter
a When an invalid parameter value is received, it seems desirable to place the invalid value(s) in ExceptionText string(s) associated with the InvalidParameterValue value.		

We assume most specific OWSs will need to specify additional allowed exceptionCode values. In addition to the standard exceptionCode values specified in each Implementation Specification, each server implementation is allowed to specify additional exceptionCode values and their meanings, for each implemented operation. These additional exceptionCode values and their meanings should be clearly documented.

Because a client may not always know what set of exceptionCode values are being used by a server, all clients should be coded to allow exceptionCode values that it does not recognize.

8.4 "locator" parameter values

Each Implementation Specification shall also specify the expected contents of the "locator" parameter value for each allowed exceptionCode, as needed for each operation specified for that OWS. Inclusion of the "locator" parameter in an Exception element shall be optional, but is recommended whenever useful information is available.

The standard contents of the “locator” parameter for each exceptionCode should be as specified in the right column of Table 19. As shown for several exceptionCodes, the “locator” parameter should be omitted when no appropriate value is defined.

EXAMPLE When the operation request includes values of a "handle" parameter, the "locator" parameter for some specified exceptionCode(s) should be the relevant value of the "handle" parameter.

8.5 Exception report XML encoding

The XML schema for an Exception Report shall be as specified in the attached owsExceptionReport.xsd file.

An example of an Exception Report encoded in XML is:

```
<?xml version="1.0" encoding="UTF-8"?>
<ExceptionReport xmlns="http://www.opengis.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/ows owsCommon.xsd"
version="1.0.0" language="en">
   <!-- Simple example. Primary editor: Arliss Whiteside. Last updated
2004/1/11. -->
   <Exception exceptionCode="MissingParameterValue" locator="service"/>
   <Exception exceptionCode="InvalidParameterValue" locator="version"/>
</ExceptionReport>
```

9 All operations except GetCapabilities, minimum abilities

9.1 Introduction

This clause specifies minimum abilities of all operations except GetCapabilities that are implemented by any OWS.

9.2 Operation request

9.2.1 Operation request parameters

A request to perform any operation except GetCapabilities shall include, in addition to operation-specific parameters, the parameters specified in Table 20.

Table 20 — Parameters used by all operation requests except GetCapabilities

Name	Definition	Data type and value	Multiplicity
service	Service type identifier	Character String type, not empty Value is OWS type abbreviation (e.g., “WMS”, “WFS”)	One (mandatory)
request	Operation name	Character String type, not empty Value is operation name (e.g., “GetCapabilities”)	One (mandatory)
version	Specification version for operation	Character String type, not empty Value format is x.y.z, where x, y, and z are non-negative integers separated by decimal points (e.g., “2.1.3”) Value is specified by each Implementation Specification and Schemas version	One (mandatory)

9.2.2 Operation request KVP encoding example

An example of a corresponding partial operation request message encoded using KVP is:

```
http://hostname:port/path?SERVICE=WCS&REQUEST=GetCoverage&VERSION
   =1.0.0&
```

9.2.3 Operation request XML encoding

A XML Schema fragment for encoding the parameters used by all operation requests except GetCapabilities is:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.opengis.net/ows"
xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:ows="http://www.opengis.net/ows" elementFormDefault="qualified"
version="0.1.4" xml:lang="en">
   <annotation>
      <appinfo>owsRequestBase.xsd 2004/4/07</appinfo>
      <documentation>
         <scope>OGC Web Service. </scope>
         <description>This XML Schema fragment encodes the common
parameters in all OWS operation requests except GetCapabilities. This
fragment should be copied and edited by each OWS, to specify a specific
value for the "service" and "version" attributes. That fragment should
then be extended for each operation used by that OWS. Primary editor:
Arliss Whiteside. </description>
         <copyright>Copyright (c) 2004 OpenGIS, All Rights
Reserved.</copyright>
      </documentation>
   </annotation>
   <!-- ============================================================
      elements and types
   ============================================================= -->
   <complexType name="RequestBaseType">
      <annotation>
```

```
         <documentation>XML encoded operation request base, for all
operations except Get Capabilities. In this XML encoding, no "request"
parameter is included, since the final element name will specify the
specific operation. </documentation>
      </annotation>
      <sequence/>
      <attribute name="service" type="string" use="required">
         <annotation>
            <documentation>Service type identifier, where the string
value is the OWS type abbreviation, such as "WMS" or "WFS".
</documentation>
         </annotation>
      </attribute>
      <attribute name="version" type="string" use="required">
         <annotation>
            <documentation>Specification version for OWS version and
operation. See Version parameter Subclause 7.3.1 for more information.
</documentation>
         </annotation>
      </attribute>
   </complexType>
</schema>
```

Each specific OWS Implementation Specification that normatively references this subclause should define a XML Schema fragment that defines a WXSRequestBaseType like the above fragment, but with the required specific values of the "service" and "version" attributes. This should be done by copying and editing the above XML Schema fragment, to specify the proper “fixed” values of the “service” and “version” attributes. This WXSRequestBaseType should then be extended to produce the complexType for each operation request.

9.3 Operation response

In the event that an OWS server encounters an error servicing an operation request, it shall return an exception report message as specified in Clause 8. The allowed exception codes shall be specified for each operation in the Implementations specification, and shall include the relevant standard exception codes listed in Table 19.

10 Other operation parameters

10.1 Introduction

This clause specifies some other parameters used in multiple OWSs by multiple operation requests and responses, including:

a) Bounding boxes

b) Coordinate reference system references

10.2 Bounding box

10.2.1 General bounding box parameters

A general bounding box is one type of bounding box that can be used by various operations in various OWSs. This very general bounding box data type is adapted from gml:EnvelopeType in the GML 3.1 draft [OGC 03-105r1]. Each general bounding box data structure shall contain the parameters specified in Table 21.

Table 21 — Parameters included in general BoundingBox data type

Name	Definition	Data type	Multiplicity and use
Lower Corner	Coordinates of bounding box corner at which the value of each coordinate normally is the algebraic minimum within this bounding box [a]	Ordered sequence of double values [b]	One (mandatory)
Upper Corner	Coordinates of bounding box corner at which the value of each coordinate normally is the algebraic maximum within this bounding box [a]	Ordered sequence of double values [b]	One (mandatory)
crs	Reference to definition of the CRS used by the LowerCorner and UpperCorner coordinates	URI	Zero or one (optional) Omitted only when specified elsewhere
dimensions	The number of dimensions in this CRS (the length of a coordinate sequence)	Positive integer	Zero or one (optional) [c]

[a] Values other that the minimum and maximum may be used as discussed in Subclauses 10.2.4 and C.13.

[b] Any number of axes can be used, from 1D to 4D or more. The number of axes included, and the order of these axes, shall be as specified by the referenced CRS.

[c] This number is specified by the CRS definition, but can also be specified here.

10.2.2 WGS 84 bounding box parameters

A WGS 84 bounding box is another type of bounding box that is expected to be used by various operations in various OWSs. This type is simplified from the general bounding box data type defined in Subclause 10.2.1, for use only with the 2D geographic coordinate reference system which uses the WGS 84 geodetic datum, where longitude precedes latitude and both are recorded in decimal degrees. Each WGS 84 bounding box data structure shall contain the parameters specified in Table 22.

Table 22 — Parameters included in WGS84BoundingBox data type

Name	Definition	Data type	Multiplicity and use
Lower Corner	Coordinates of bounding box corner at which the values of latitude and longitude normally are the algebraic minimums within this bounding box [a]	Ordered sequence of two double values in decimal degrees, with longitude before latitude	One (mandatory)
Upper Corner	Coordinates of bounding box corner at which the values of latitude and longitude normally are the algebraic maximums within this bounding box [a]	Ordered sequence of two double values in decimal degrees, with longitude before latitude	One (mandatory)
crs	Reference to definition of the CRS used by the LowerCorner and UpperCorner coordinates	URI [b]	Zero or one (optional) Included when expected to be useful
dimensions	The number of dimensions in this CRS (the length of a coordinate sequence)	Positive integer Value = 2	Zero or one (optional) [c]

[a] Values other that the minimum and maximum may be used as discussed in Subclauses 10.2.5 and C.13.

[b] Reference to 2D CRS using WGS 84 datum with longitude before latitude in decimal degrees, as specified in Subclause 10.3.

[c] The number “2” is specified by the WGS 84 2D CRS definition, but can also be specified here.

10.2.3 Bounding box KVP encoding

The general bounding box parameters shall be KVP encoded as specified in Subclause 11.5.3 for a parameter value containing an (ordered) list. For a general bounding box, the listed values shall be for the ordered quantities:

LowerCorner coordinate 1
LowerCorner coordinate 2
LowerCorner coordinate 3
...
LowerCorner coordinate N
UpperCorner coordinate 1
UpperCorner coordinate 2
UpperCorner coordinate 3
...
UpperCorner coordinate N
CRS URI (optional)

This list allows N coordinates for each corner, listed in the order specified by the associated CRS. The exact number of coordinates specified by the associated CRS shall be included. A parser can determine the number of dimensions, and whether or not the optional CRS URI is present, by counting the number of items in the list. If there are an odd number of items, then a CRS URI is present. The number of remaining items divided by two indicates the number of dimensions of the bounding box.

The CRS URI value usually references an instance of the definition of a CRS, as specified in [OGC Topic 2]. Such a CRS definition can be XML encoded using the gml:CoordinateReferenceSystemType in [GML 3.1]. For well known references, it is not required that the CRS definition exists at the location the URI points to. If no CRS URI value is included, the applicable CRS must be either:

a) Specified outside the bounding box, but inside a data structure that includes this bounding box, as specified for a specific OWS use of this bounding box type

b) Fixed and specified in the Implementation Specification for a specific OWS use of the bounding box type

A WGS 84 bounding box shall be KVP encoded in a corresponding parameter value list, with the ordered listed values for the quantities:

LowerCorner longitude, in decimal degrees
LowerCorner latitude, in decimal degrees
UpperCorner longitude, in decimal degrees
UpperCorner latitude, in decimal degrees
CRS URI = “urn:opengis:def:crs:OGC::84” (optional)

The CRS URI can be included when considered useful. When included, this CRS URI shall reference the 2D WGS 84 coordinate reference system with longitude before latitude and decimal values of longitude and latitude, using the value listed above. Again, it is not required that the CRS definition exists at the location the URI points to.

Both types of bounding box parameters shall use application-specific parameter names which are not specified here, but shall be specified for each bounding box used in each specific OWS Implementation Specification. An Implementation Specification can specify a KVP encoding of an operation request that contains more than one bounding box parameter. An Implementation Specification shall specify when a KVP encoded bounding box is limited to, or shall be interpreted as, a WGS 84 bounding box.

Examples of KVP encoded bounding boxes are:

```
VWXYZWGS84BOX=71.63,41.75,-70.78,42.90

ABCDEBOX=189000,834000,285000,962000,urn:opengis:def:crs:OGC::84
```

NOTE In the second example, the colons in the CRS URI must be escaped by “%3A”, as required in KVP encoding. The resulting encoding is “ABCDEBOX=189000,834000,285000,962000, urn%3Aopengis%3Acrs%3AOGC%3A%3A84”. See 11.3 for more on reserved characters and URL encoding.

10.2.4 Bounding box XML encoding

The XML Schema fragment for encoding either a general or a WGS 84 bounding box shall be as specified in the attached owsBoundingBox.xsd file. Notice that this XML Schema defines both BoundingBoxType and WGS84BoundingBoxType, which should be used as the types of XML elements with application-specific names other than BoundingBox and WGS84BoundingBox.

Two examples of XML encoded bounding boxes are:

```
<BoundingBox crs="urn:opengis:def:crs:EPSG::26986" dimensions="2">
   <LowerCorner>189000 834000</LowerCorner>
   <UpperCorner>285000 962000</UpperCorner>
</BoundingBox>

<WGS84BoundingBox>
   <LowerCorner>-71.63 41.75</LowerCorner>
   <UpperCorner>-70.78 42.90</UpperCorner>
</WGS84BoundingBox>
```

10.2.5 Bounding box use

Bounding boxes can be repeated wherever useful in a specific OWS Implementation Specification. Wherever a specific OWS allows the bounding box to be repeated, that Implementation Specification shall specify how multiple bounding boxes shall be interpreted, by OWS clients and/or servers. One expected use is meaning the union of the areas defined by multiple listed bounding boxes. That meaning is expected to be often useful in describing the region(s) covered by geospatial data sets.

The coordinates of each bounding box corner are normally the algebraic minimum and maximum inclusive values of the position coordinates of all the data within this bounding box. For features, these minimum and maximum values can be computed from the positions of all the points in all included geometries (in the referenced CRS). For grid coverages, these values can be computed from the positions of the areas covered by all the grid cells (or image pixels).

The general bounding box contents defined will not always specify the MINIMUM rectangular BOUNDING region, if the referenced CRS uses an Ellipsoidal, Spherical, Polar, or Cylindrical coordinate system, as those terms are specified in OGC Abstract Specification Topic 2. Specifically, this box will not specify the minimum rectangular bounding region surrounding a geometry whose set of points span the value discontinuity in an angular coordinate axis. Such axes include the longitude and latitude of Ellipsoidal and Spherical coordinate systems. That geometry could lie within a small region on the surface of the ellipsoid or sphere.

If the data for which a bounding box is needed is continuous around the continuous angular axis of an Ellipsoidal, Spherical, Polar, or Cylindrical coordinate system, the bounding box limits for that angular axis shall be set to minus and plus infinity.

EDITOR’S NOTE The Harmonization working group decided to NOT NOW specify a bounding box structure that can always specify the MINIMUM rectangular region SURROUNDING data within a limited region that crosses a value discontinuity. The following paragraph thus specifies that each specific OWS Implementation Specification shall suitably address this issue.

For each use of the a bounding box data structure, a specific OWS Implementation Specification should specify if that use shall allow specifying the minimum rectangular bounding region for data within a limited region that crosses a value discontinuity for some (or all) allowed CRSs. If the minimum rectangular bounding region shall be allowed for some CRSs, that specific OWS Implementation Specification shall also specify how that can be done when the referenced CRS allowed uses an Ellipsoidal, Spherical, Polar, or Cylindrical coordinate system.

There are a variety of possible approaches to allowing specification of the minimum rectangular bounding region when the referenced CRS uses an Ellipsoidal, Spherical, Polar, or Cylindrical coordinate system. Subclause C.13 (informative) summarizes the known alternatives for handling the case where the minimum region crosses the value discontinuity in a longitude or other continuous axis, and recommends the first two listed alternatives.

10.3 Coordinate reference system references

10.3.1 Overview

This subclause specifies a group of alternative ways to reference a CRS, in OWS operation requests and responses. One frequent use will be referencing the CRS for a bounding box. In most cases, these ways will be used to identify the referenced CRS, and not to transfer a definition of that CRS. Subclause C.14 summarizes many of the requirements considered when specifying how to reference CRSs. Much of this material is also applicable to referencing CRS components and Coordinate Operations and their components.

A specific OWS shall always reference a CRS by using an XML attribute or element with the type anyURI. Such an anyURI value can be used to reference a CRS whether the definition of that CRS is included in the same data transfer, is NOT included in the same data transfer, cannot be electronically accessed, or can be electronically accessed.

NOTE XML attributes with type anyURI include the GML 3.1 defined attributes named gml:srsName, gml:uom, xlink:href, and gml:codeSpace.

When using a XML attribute or element with the type anyURI to reference a CRS or CRS-related object, that URI shall have a value which uses one of two alternative URI formats:

a) Universal Resource Locator (URL), with standard form. The URL format should be used whenever the referenced definition is known to be electronically available using this standard URL.

b) Universal Resource Name (URN), with a specified form. The URN format shall be used whenever the referenced definition is not, or might not be, available using a URL. This URN shall reference data that is specified by some “authority” and is “well-known” to both client and server software, including multiple clients and multiple servers.

Use of URNs is expected to be more common than use of URLs, and specific OWS Implementation Specifications are expected to specify many standard URN values.

10.3.2 URL references

For all XML attributes and elements with the anyURI data type, a URL value can be used, and often should be used, to reference a definition that is known to be always available using this URL. When not in the same UML document, those definitions shall be electronically available over the Internet using this URL, to both client and server software including multiple clients and multiple servers that must interoperate. The available definitions shall be encoded in XML, using one or more Application Schemas based on the CRS Schemas in Clause 12 of [GML 3.1].

Such a URL value shall reference either a:

a) Document that defines only the referenced object, optionally including definitions of all or some of its components

b) Dictionary document containing multiple objects, also referencing the specific object within that dictionary using its gml:id value

c) Web service that stores definitions, where the anyURI value references a GetXxxx operation and provides all the operation parameters needed to retrieve the referenced object

d) Elsewhere in the same XML document, either in a:

   1) Logically appropriate place defined by an XML Schema

   2) Metadata element encoded within an XML element that includes all the references to those object definitions, such as the outer-most element of the XML document

EDITOR’S NOTE The Harmonization working group decided to NOT NOW define how a server shall use such a URL in order to be considered compliant with an OWS Implementation Specification. The following paragraph thus specifies that each specific OWS Implementation Specification shall suitably address this issue.

Wherever a specific OWS Implementation Specification allows such a URL to be used, it shall specify how servers shall use those URLs in order to be considered compliant with that specification. There are several alternative approaches to compliant server uses of such transferred URLs, as briefly described in Subclause C.14.6.

10.3.3 URN format for single objects

When referencing a single object, a URN value for an anyURI data type shall have the form:

urn:opengis:def:objectType:authority:version:code

The “urn”, “opengis”, “def”, and six “:” parts of this URN are fixed. The “opengis” part shall be a registered namespace authority for all URNs used by the OGC. The “def” part shall be the fixed category label which identifies all the OGC URNs that reference object definitions. In this use, all textual parts of URN values shall be case-insensitive.

The “objectType” part shall be a unique identifier of the type of the referenced definition. The “authority” part shall be the OGC-specified abbreviation for the authority organization that specified the referenced definition. The optional “version” part shall be the version of the authority or code for the referenced definition. The “code” part shall be a unique identifier of the referenced definition, as specified by the referenced authority. The “code” part can be human-understandable, provided that it is unique for that authority and version.

EDITOR’S NOTE This and the following URN forms are still subject to change, in order to harmonize these definition URN forms with other OGC uses of the ”opengis” namespace authority. In particular, the “def” category label might be changed in this harmonization.

NOTE 1 The ”opengis” part denotes the namespace authority in a URN, and the value used should be registered with IANA. Until the ”opengis” or other value is so registered by the OGC, the “x-opengis” value should be used, where the “x” denotes an experimental namespace.

NOTE 2 The ”authority” part denotes an authority recognized by the OGC. We here specify that the OGC recognize the European Petroleum Survey Group as an authority, with the abbreviation “EPSG”. We here assume that the OGC will also recognize itself as an authority, here using “OGC” as its abbreviation. The OGC must specify the “code” values that defined by that “OGC” authority, which we assume will be done in separate OGC specification(s).

The “version” part of a URN shall be omitted when the referenced definition does not have a version, and the referenced definition is not specific to an authority version. When included, the “version” shall be recorded in the format specified by the authority, sometimes “N.N.N” or “N.N”, where each “N” stands for an integer. No "v" or other version prefix shall be included.

The “objectType” part is required, and shall contain the name of the XML element with type gml:IdentifierType that should contain the referenced “code” value, when this definition is encoded in XML. However, the “ID” suffix in that element name shall be omitted. For the types of GML objects specified in Clause 12 of [GML 3.1], those XML elements are named, deleting the “ID” suffix:

a) crs, for all coordinate reference systems

EDITOR’S NOTE This identifier has been changed from “srs”, and I have proposed that this be changed in [OGC Topic 2] and Clause 12 of [GML 3.1].

b) datum, for all datums

c) meridian, for all prime meridians

d) ellipsoid, for all ellipsoids

e) cs, for all coordinate systems

f) axis, for all coordinate system axes

g) coordinateOperation, for all coordinate operations

EDITOR'S NOTE The coordinateOperation identifier might be abbreviated to "operation" or "transformation".

h) method, for all operation methods

i) parameter, for all operation parameters

j) group, for all operation parameter groups

k) uom, for all units of measure

NOTE 3 The uom identifier listed last above is not specified in [GML 3.1] or elsewhere. In this case, the referenced "code" value should be the one in the catalogSymbol element, in the UnitDefinitionType defined in the units.xsd schema of [GML 3.1].

Specializing the above, the URN value for an anyURI data type that references one object in the European Petroleum Survey Group (EPSG) database shall have the form:

urn:opengis:def:objectType:EPSG:version:code

In this case, the "authority" part of a URN shall be "EPSG". In this case, the "version" part is optional, since the EPSG doesn't currently change existing definitions. The "code" part of a URN should be the EPSG "code" unique identifier of the referenced definition. Alternately, the "code" part of a URN can be the EPSG "name" unique identifier.

An example URN value for CRS 26986 specified by the EPSG is:

```
urn:opengis:def:crs:EPSG::26986
```

10.3.4 URN format for not-completely-specified objects

A URN can also be used to reference a not-completely-specified CRS or other object, which is missing the values of a few identified parameters. When referencing an object with two unspecified parameters, a URN value for an anyURI data type shall have the form:

urn:opengis:def:objectType:authority:version:code:value1:value2

In this URN form, the values for the previously-identified parameters are added to the URN form for a (completely specified) single object, as specified in Subclause 10.3.3. The obvious variations on the URI form can be used when one or three parameters are unspecified. The object identified by the "authority", "version", and "code" must be completely specified with the exception of the values for a few identified parameters.

Whenever such a parameter value requires a unit of measure (uom), the uom for that parameter shall be specified by the referenced not-completely-specified object.

An example URN value for the Auto Orthographic CRS 42003 specified in Annex B of WMS 1.3.0 is:

```
urn:opengis:def:crs:OGC::42003:1:-100:45
```

10.3.5 URN references to combined objects

In some cases, it is useful to reference two or more well-known CRS-related objects that are combined. This shall be done by concatenating the object references in one URN. That is, a URN can concatenate the URNs of the two individual well-known objects, using the URN form:

> urn:opengis:def:objectType,objectType:authority:version:code,
> objectType:authority:version:code

Similarly, a URN can concatenate the URNs of the three individual well-known objects, using the URN form:

> urn:opengis:def:objectType,objectType:authority:version:code,
> objectType:authority:version: code,objectType:authority:version:code

NOTE When URNs are concatenated like this, the combined object is implicitly defined, and is not assigned a separate xxxName or xxxID. (That is, the defined object is anonymous.) Combined references should not be used when the defined object is well-known.

In a concatenated URN, the first “objectType” is the type of the combined object. Combined references in a URN can be used for defining several types of objects, including for:

a) Compound coordinate reference systems
b) Combining a datum and a coordinate system into a coordinate reference system
c) Projected or derived coordinate reference systems
d) Concatenated operations
e) Objects defined by specified Application Schemas

These first four combinations listed above are described in the following subclauses.

10.3.6 URN combined references for compound coordinate reference systems

A URN reference to combined objects can be allowed for any compound coordinate reference system (CompoundCRS) that combines two or three well-known CRSs. This alternative is allowed when there is not a well-known CompoundCRS that combines these specific well-known (non-compound) CRSs. In this case, the URN shall concatenate the URNs of the two or three individual well-known CRSs. Of course, the referenced CRSs shall be suitable for combination in one CompoundCRS.

The URNs of the individual well-known CRSs shall be listed in the order in which the coordinate axes are combined in a coordinate value. All the “objectType” values shall be “crs”.

10.3.7 URN combined references for datum and coordinate system

A URN reference to combined objects can be allowed for a coordinate reference system that combines a well-known datum with a well-known coordinate system. This alternative is allowed when there is not a well-known coordinate reference system that combines these specific well-known datum and coordinate system. In this case, the URN shall concatenate the URNs of one well-known datum and one well-known coordinate system. The referenced datum can be of any concrete subtype, and the referenced coordinate system can be of any concrete subtype applicable to that specific datum.

The URNs of the individual components shall be listed in the order of datum and then coordinate system. The three “objectType” values shall be “crs”, “datum” and “cs”.

EDITOR’S NOTE The EPSG may define a different method to effectively combine an EPSG datum and an EPSG coordinate system, by just concatenating two EPSG codes. Is this true?

10.3.8 URN combined references for projected or derived CRSs

A URN reference to combined objects can be allowed for any projected coordinate reference system (ProjectedCRS) that combines a well-known GeographicCRS and a well-known (defined by) Conversion. This alternative is allowed when there is not a well-known ProjectedCRS that combines these specific well-known GeographicCRS and Conversion. In this case, the URN shall concatenate the URNs of the one well-known CRS, one well-known Conversion, and one well-known CartesianCS. Of course, the referenced Conversion must be a map projection applicable to a GeographicCRS, and must produce a CartesianCS.

Similarly, a URN reference to combined objects can be allowed for any derived coordinate reference system (DerivedCRS) that combines a well-known base CRS, a well-known CoordinateSystem, and a well-known (defined by) Conversion.

The URNs of the individual components shall be listed in the order of GeographicCRS, CartesianCS, and then Conversion. The four “objectType” values shall be “crs”, “crs”, “cs”, and “coordinateOperation”.

10.3.9 URN combined references for concatenated operations

A URN reference to combined objects can be allowed for any concatenated coordinate operation (ConcatenatedOperation) that combines two or more well-known coordinate operations. This alternative is allowed when there is not a well-known ConcatenatedOperation that concatenates these specific coordinate operations. In this case, the URN shall concatenate the URNs of the two or more well-known coordinate operations. The referenced coordinate operations can be of any concrete subtype including ConcatenatedOperation. Of course, the referenced CRSs shall be suitable for combination in one ConcatenatedOperation.

NOTE For example, the coordinate operations concatenated must meet the stated constraint: The sequence of operations is constrained by the requirement that the source coordinate reference system of step (n+1) must be the same as the target coordinate reference system of step (n). The source coordinate reference system of the first step and the target coordinate reference system of the last step are the source and target coordinate reference system associated with the concatenated operation.

The URNs of the individual coordinate operations shall be concatenated in the order of coordinate operation application. The “objectType” values shall all be “coordinateOperation”.

11 Operation request and response encoding

11.1 General HTTP rules

This specification applies to OWS interfaces that use the distributed computing platform (DCP) comprising Internet hosts that support the Hypertext Transfer Protocol (HTTP) [IETF RFC 2616]. Thus the Online Resource of each operation supported by a server is an HTTP Uniform Resource Locator (URL). Each URL shall conform to the description in Section 3.2.2 "HTTP URL" of [IETF RFC 2616], but is otherwise server implementation dependent. Only the query portion comprising the service request itself is defined by this specification.

This document defines two methods of encoding OWS operation requests. One method uses XML as the encoding language, and the other method uses keyword-value pairs to encode the various parameters. An example of a keyword value pair (KVP) is:

```
REQUEST=GetCapabilities
```

where "REQUEST" is the parameter name and "GetCapabilitites" is the value. A KVP value can include XML encoded elements. In both cases, the response to a request or exception reporting must be identical.

HTTP supports two request methods: GET and POST. One or both of these methods may be offered by a server for each operation, and the use of the Online Resource URL differs in each case.

11.2 HTTP GET

An Online Resource URL intended for HTTP GET requests is in fact only a URL prefix to which additional parameters are appended in order to construct a valid operation request. A URL prefix is defined in accordance with [IETF RFC 2396] as a string including, in order, the scheme ("http" or "https"), Internet Protocol hostname or numeric address, optional port number, path, mandatory question mark '?', and optional string comprising one or more server-specific parameters ending in a mandatory ampersand '&'. The prefix defines the network address to which request messages are to be sent for a particular operation on a particular server, and may also identify a configuration of that server. Each operation may have a different prefix, and each prefix is entirely at the discretion of the service provider.

This document defines how to construct a query part that is appended to the URL prefix in order to form a complete request message. Every OWS operation request has several mandatory and/or optional request parameters. Each parameter has a defined name. Each parameter may have one or more legal values, which are either defined by this document, defined by an Implementation Specification that builds on this document, or are selected by the client based on service metadata. To formulate the query part of the URL, a client shall append the mandatory request parameters, and any desired optional parameters, as name/value pairs in the form "name=value&" (parameter name, equals sign, parameter value, ampersand). The '&' is a separator between name/value pairs, and is therefore optional after the last pair in the request string.

When the HTTP GET method is used, the client-constructed query part is appended to the URL prefix defined by the server, and the resulting complete URL is invoked as defined by HTTP [IETF RFC 2616]. Table 23 summarizes the components of an operation request URL when HTTP GET is used.

Table 23 — Structure of operation request using HTTP GET

URL component	Description
http://host[:port]/path[?{name[=value]&}]	URL prefix of service operation. [ ] denotes 0 or 1 occurrence of an optional part; {} denotes 0 or more occurrences.
name=value&	One or more standard request parameter name/value pairs as defined for each operation by this International Standard. This parameter encoding is referred to as Keyword Value Pair (KVP) encoding in this document.

11.3 Reserved and encoded characters in HTTP GET URLs

The URL specification [IETF RFC 2396] states that all characters other than:

a) Reserved characters being used for their defined purpose,

b) Alphanumeric characters, and

c) The characters "-", "_", ".", "!", "~", "*", "'", "(", and ")"

shall be encoded as "%xx", where xx is the two hexadecimal digits representing the octet code of the character. Within the query string portion of a URL (i.e., everything after the "?"), the space character (" ") is an exception, and shall be encoded as a plus sign ("+"). A server shall be prepared to decode any character encoded in this manner.

This specification explicitly reserves several characters for use in operation requests. When the characters '&', '=', ',' and '+' appear in one of the roles defined in Table 24, they shall appear literally in the URL. When those characters appear elsewhere (for example, in the value of a parameter), they shall be encoded as defined in [IETF RFC 2396].

Table 24 — Reserved characters in operation request strings

Character	Reserved usage
?	Separator indicating start of query string.
&	Separator between parameters in query string.
=	Separator between name and value of parameter.
,	Separator between individual values in list-oriented parameters (such as BBOX, LAYERS and STYLES in the WMS GetMap request).
+	Shorthand representation for a space character in the query string.

11.4 HTTP POST

An Online Resource URL intended for HTTP POST operation requests is a complete URL (not merely a prefix as in the HTTP GET case) that is valid according to [IETF RFC 2396]. This is the URL to which clients transmit request parameters in the body of the POST message. An OWS shall not require additional parameters to be appended to the URL in order to construct a valid target for the operation request. The URL may be different for each operation, or the same, at the discretion of the server provider. When POST is used, the operation request message can be encoded as an XML document, formatted as specified by one or more XML Schemas. When POST is used, the operation request message can alternately be KVP encoded.

11.5 KVP encoding

11.5.1 Introduction

This subclause specifies the Keyword Value Pair (KVP) encoding of the operation request parameters specified in Clauses 7 through 10.

KVP encoded values can include XML encoded data, with that data encoded in one XML element.

11.5.2 Capitalization

The capitalization of parameter names when KVP encoded shall be case insensitive, meaning that parameter names may have mixed case or not.

EXAMPLES The “request” parameter name could be REQUEST, request, Request, or ReQuEsT.

NOTE The XML capitalization is uniformly used in Clauses 7 through 10 plus Annex C of this document.

The capitalization of parameter values when encoded using Keyword Value Pairs shall be as used in Clause 7 through 10 of this document. More generally, all value strings shall have the first word and any subsequent words in the name capitalized. All other letters will be lower case.

EXAMPLE One possible “request” parameter value is GetCapabilities.

11.5.3 Parameter value lists

Parameters values containing lists (for example, AcceptVersions and AcceptFormats in the GetCapabilities operation request) shall use the comma (",") as the separator between items in the list. Additional white space shall not be used to delimit list items. If a list item value includes a space or comma, it shall be escaped using the URL encoding rules [IETF RFC 2396].

In some lists, individual entries may be empty, and shall be represented by the empty string (""). Thus, two successive commas indicates an empty item, as does a leading comma or a trailing comma. An empty list ("") can either be interpreted as a list containing no items or as a list containing a single empty item, depending on the context.

11.5.4 Numeric and boolean values

Integer numbers shall be represented in a manner consistent with the specification for integers in Section 3.3.13 of [XML Schema Part 2: Datatypes]. Each Implementation Specification shall explicitly specify where an integer value is mandatory.

Real numbers shall be represented in a manner consistent with the specification for double-precision numbers in Section 3.2.5 of [XML Schema Part 2: Datatypes]. This representation allows for integer, decimal and exponential notations. This representation also allows special values representing infinity and not-a-number, which shall not be used except where specifically allowed by an Implementation Specification. A real value is allowed in all numeric fields unless the value is explicitly restricted to integer.

Boolean values shall be represented by the uppercase strings "TRUE" and "FALSE", representing Boolean true and false respectively. Each Implementation Specification shall explicitly specify where a Boolean value is mandatory.

11.6 XML encoding

11.6.1 Introduction

This subclause specifies the XML encoding of the operation request and response parameters specified in Clauses 7 through 10.

11.6.2 Capitalization

The capitalization of parameter and operation names when encoded as XML elements and attributes shall be as used in Clauses 7 through 10 of this document. More generally, these name capitalization rules shall be used:

a) All names of XML elements shall have the first word and any subsequent words in the name capitalized. All other letters will be lower case.

b) All names of XML attributes shall have the first word in lower case and any subsequent word in the name capitalized. All other letters shall be lower case.

EXAMPLES The GetCapabilities operation request element name shall be GetCapabilities. The updateSequence attribute name shall be updateSequence.

The capitalization of parameter values when encoded as XML stings shall be as used in Clauses 7 through 10 of this document. More generally, all XML string values shall have the first word and any subsequent words in the name capitalized. All other letters will be lower case.

EXAMPLE One possible "request" attribute value is GetCapabilities.

11.6.3 XML Schema documentation

All XML Schemas shall contain documentation of the meaning of each element, attribute, and type. In many cases, a documentation element is included only for the (complex or simple) types, but is applicable to all the elements or attributes that use that type. All of these documentation elements shall be considered normative, except where labelled "informative".

11.6.4 Namespaces

Namespaces are used to discriminate XML elements, attributes, and data types defined in application-specific domains from one another [Namespaces In XML]. Multiple normative XML namespaces definitions are thus used is different XML Schemas for each OWS, and shall be suitably implemented by each compliant OWS server. While many of the examples in this document and OWS interface specifications use a single namespace, multiple namespaces shall be supported by each compliant OWS server.

11.6.5 XML Schema extension and restriction

The XML Schemas specified in this document will usually require extension for use by a specific OWS, and may require restriction and/or subsetting for use by a specific OWS. Such extension, restriction, and/or subsetting should be done in a manner similar to development of GML Application Schemas as described in Subclause 6.2 and Clause 23 of [GML 3.1]. The following subclauses provide some more specific information about Application Schemas.

Some Application Schemas and other uses of these OWS Schemas will use only a subset of the XML elements, types, and other capabilities defined herein. Such a subset is termed a Profile in GML, as specified in Clause 22 of the GML 3.1 Implementation Specification. Briefly, a profile is a specified subset of the elements, types, etc. defined in these XML Schemas, often selected to improve interoperability and reduce ambiguity. Such a profile should be specified by an Application Schema.

11.6.6 Application schemas

Most of the concrete XML elements defined in these OWS Schemas can be used without Application Schemas, whenever no content extensions or restrictions are needed. An Application Schema shall be used whenever element contents extension is required, and should be used in most other cases to specify needed restrictions. That is, an Application Schema should be defined to extend and/or restrict XML elements as needed for a specific OWS, to:

a) Add elements to contents of existing elements, for recording additional data about that item needed for that OWS application.

b) Restrict the multiplicity of current contents elements, to eliminate flexibility not needed and perhaps confusing for that OWS application.

c) Use a different element name, to be more easily understood in that specific OWS application, primarily for elements that will be instantiated many times.

d) Specify standard contents and contents patterns for selected elements and attributes, as needed to improve interoperability.

e) Specify standard XML and other documents to be referenced or otherwise used, as needed to improve interoperability.

Application Schemas can be used for XML document contents extensions, restrictions, or both. Contents extension is expected to be often used to record additional data needed for applications. Contents restriction is expected to be frequently used to restrict contents, in order to increase interoperability and reduce ambiguity when greater flexibility is not needed for applications.

An Application Schema is an XML Schema that imports and builds upon one or more of the OWS Schemas specified in this document. Such a Schema defines one or more XML elements useful for transfer of OWS operation responses and requests. An Application Schema can specify a single top level element for use by an XML document, with the XML elements and types that it uses. Such an Application Schema will import and build upon one or more of the OWS Schemas specified in this document.

Each application schema must declare a target namespace. This is the namespace in which the XML elements or terms of the vocabulary “live”. This shall not be the OWS or GML namespace. A target namespace is declared in the application schema using the targetNamespace attribute of the schema element from XML Schema.

Each Application Schema must import the necessary XML Schemas specified in this document, and perhaps from GML 3, with the correct namespace assignment. For example, in order to define coordinate reference systems, it was necessary to import coordinateReferenceSystems.xsd, either directly or indirectly. Direct import is done by including the declaration:

```
<xsd:import namespace="http://www.opengis.net/gml"
schemaLocation="../coordinateReferenceSystems.xsd"/>
```

Notice that this <import> element example specifies that the components described in coordinateReferenceSystems.xsd are in the GML namespace http://www.opengis.net/gml. This namespace identifier must match the target namespace specified in the schema being imported, to ensure XML Schema validity.

The schemaLocation of the imported .xsd file can be a local reference or a URL reference to the file. A URL reference can be to some remote repository, such as the repository http://schemas.opengis.net/ on the OGC web site. The above example assumes that the coordinateReferenceSystems.xsd file is stored locally at a location relative to this Application Schemas .xsd file.

Alternately, necessary XML Schemas can be imported indirectly, by importing another Application Schema that imports the needed Schemas. In addition, the required import of any schema may be provided by the import of an equivalent subset schema as described in Clause 22 of the GML 3.1 Implementation Specification. These are all equivalent schemas with respect to satisfying the schema import requirements.

Note that XML elements included in complex types that are defined with local names in an Application Schema will prevent derivation by restriction in another namespace, unless the local names are dropped in the restriction. Such complex types are appropriate for elements intended for use “as is” in their own namespace, and should be declared to be final=”restriction”. Elements included in complex types by reference to global elements support derivation by restriction in another namespace, allowing restriction of cardinality, and/or replacement by a member of a substitution group. Such complex types designed for derivation by restriction are appropriate “library types” for elements in substitution groups that cross namespaces.

11.7 HTTP responses

Upon receiving a valid operation request, the service shall send a response corresponding exactly to the request as detailed in the Implementation Specification, or send an exception report if unable to respond correctly. Only in the case of Version Negotiation (Subclause 7.3.2) may the server returning a differing result. Upon receiving an invalid request, the service shall issue an exception report as specified in Clause 8.

NOTE As a practical matter, a client should be prepared to receive either a valid result, or nothing, or any other result. This is because the client may have formed a non-conforming request that inadvertently triggered a reply by something other than an OWS, because the server may be non-conforming, etc.

A server may send an HTTP Redirect message (using HTTP response codes as defined in [IETF RFC 2616]) to an absolute URL that is different from the valid request URL that was sent by the client. HTTP Redirect causes the client to issue a new HTTP request for the new URL. Several redirects could in theory occur. Practically speaking, the redirect sequence ends when the server responds with an operation response. The final response shall be an OWS operation response that corresponds exactly to the original operation request (or an exception report).

Response objects shall be accompanied by the appropriate Multipurpose Internet Mail Extensions (MIME) type [IETF RFC 2045] for that object. A list of MIME types in common use on the internet is maintained by the Internet Assigned Numbers Authority [IANA]. Allowable types for operation responses and exception reports are discussed below.

The basic structure of a MIME type is a string of the form "type/subtype". MIME allows additional parameters in a string of the form "type/subtype; param1=value1; param2=value2". A server may include parameterized MIME types in its list of supported output formats. In addition to any parameterized variants, the server should offer the basic un-parameterized version of the format.

Response objects should be accompanied by other HTTP entity headers as appropriate and to the extent possible. In particular, the Expires and Last-Modified headers provide important information for caching; Content-Length may be used by clients to know when data transmission is complete and to efficiently allocate space for results, and Content-Encoding or Content-Transfer-Encoding may be necessary for proper interpretation of the results.

12 Guidance for OWS Implementation Specifications

EDITOR’S NOTE This clause needs to be expanded, in future versions of this document or in a separate document containing guidance for authors and editors.

This clause provides some guidance for editors of OWS Implementation Specifications, currently in the form of a list:

a) Rather than continuing to duplicate common material in each Implementation Specification, each specification should normatively reference each relevant part of this document. Such normative references can take the form of stating “This TBD shall include TBD as specified in Subclause TBD of OGC document TBD.”

b) The “Normative references” Clause of each Implementation Specification that normatively references any part of this document shall list this document, and shall include the relevant version number of this document.

Annex A
(normative)

XML schemas

In addition to this document, this specification includes several normative XML Schema files. These are posted online at the URL http://schemas.opengis.net/ows/. These XML Schema files are also bundled with the present document. In the event of a discrepancy between the bundled and online versions of the XML Schema files, the online files shall be considered authoritative.

The common OWS abilities now specified in this document use seven specified XML Schemas included in the zip file with this document. These XML Schema files roughly match the six UML packages described in Annex B, and are named:

owsServiceIdentification.xsd

ows19115subset.xsd

owsServiceProvider.xsd

owsOperationsMetadata.xsd

owsExceptionReport.xsd

owsBoundingBox.xsd

owsGetCapabilities.xsd

The first six XML Schema files listed above are referenced in Subclauses 7.4.6, 8.5, and 10.2.4 of this document. The attached owsGetCapabilities.xsd file specifies the combination of the other XML Schema fragments listed in Clause 7, eliminating duplications. This owsGetCapabilities.xsd XML Schema “includes” the first four XML Schema files listed above. All these XML Schemas contain documentation of the meaning of each element, attribute, and type, and this documentation shall be considered normative as specified in Subclause 11.6.3.

The RequestBase XML Schema fragment listed in Subclause 9.2.3 is currently included in the attached schemas, but is not expected to be directly used since both of its attributes have different fixed values for each specific OWS. As stated in Subclause 9.2.3, each specific OWS Implementation Specification should define a XML Schema fragment that defines a WXSRequestBaseType like that fragment, but with the required specific “fixed” values of the "service" and "version" attributes. This should be done by copying and editing this XML Schema fragment. This WXSRequestBaseType should then be extended to produce the complexType for each operation request.

Annex B
(informative)

UML models

B.1 OWS Common package

Figure B.1 shows an OWS Common UML package that includes the:

a) OWService interface class that models the GetCapabilities operation

b) GetCapabilities and Section classes that model the GetCapabilities operation request specified in Subclause 7.2

c) ServiceIdentification class that models the GetCapabilities operation response specified in Subclause 7.4. The associated ServiceIdentification, ServiceProvider, and OperationsMetadata classes shown are modeled in the OWS Service Identification, OWS Service Provider, and OWS Operations Metadata UML packages defined in Subclauses B.2 through B.4.

d) RequestBase class that models the parameters included in all operations except GetCapabilities specified in Subclause 9.2

Many of the classes in the OWS Common UML package shown in Figure B.1 are {Abstract}, because they must be specialized for each specific OWS. The classes introduced by this package are further defined by Tables 1, 6, 7, and 21 in this document.

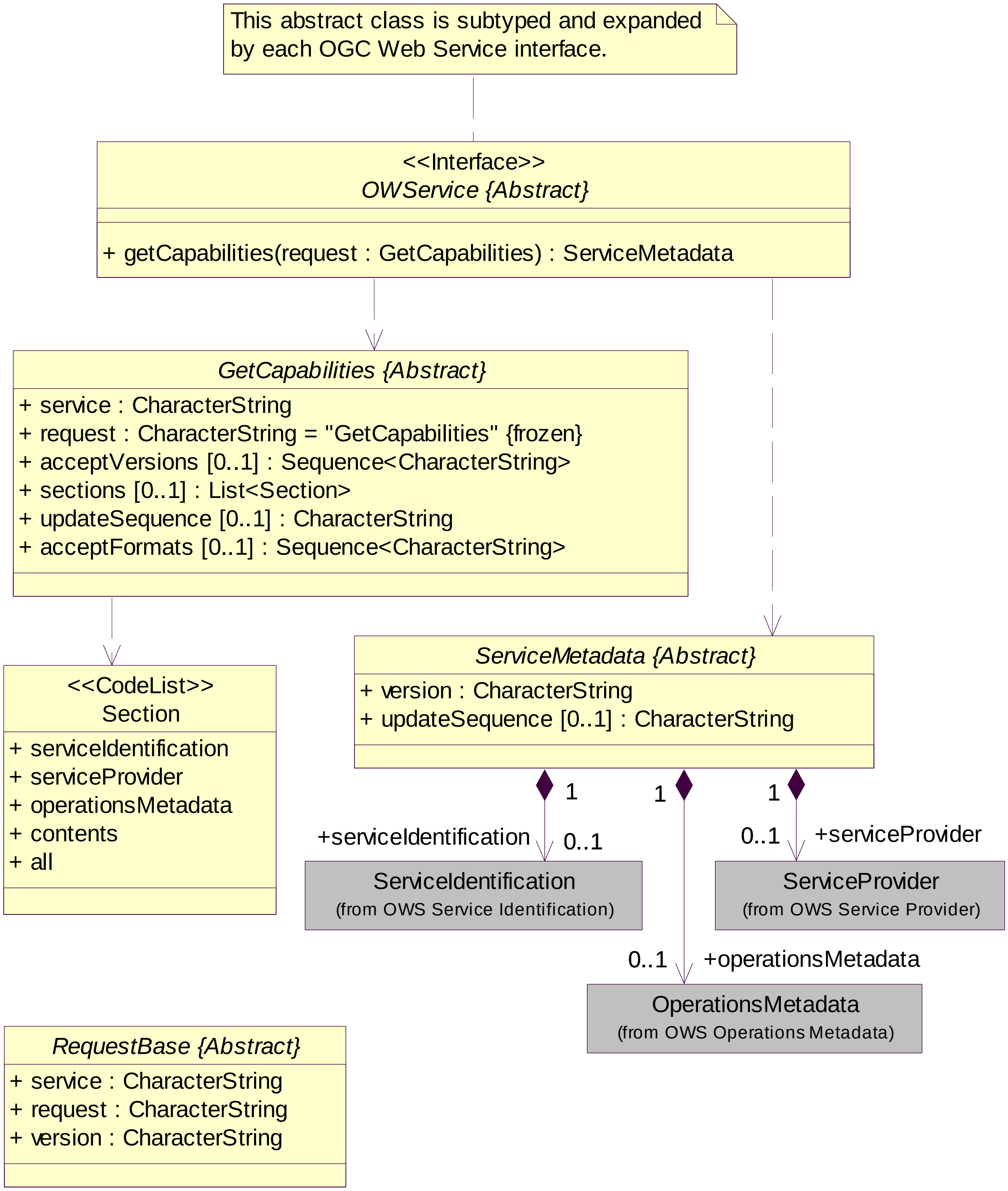
This abstract class is subtyped and expanded by each OGC Web Service interface.
<<Interface>>
OWService {Abstract}
+ getCapabilities(request : GetCapabilities) : ServiceMetadata
GetCapabilities {Abstract}
+ service : CharacterString
+ request : CharacterString = "GetCapabilities" {frozen}
+ acceptVersions [0..1] : Sequence<CharacterString>
+ sections [0..1] : List<Section>
+ updateSequence [0..1] : CharacterString
+ acceptFormats [0..1] : Sequence<CharacterString>
<<CodeList>>
Section
+ serviceIdentification
+ serviceProvider
+ operationsMetadata
+ contents
+ all
ServiceMetadata {Abstract}
+ version : CharacterString
+ updateSequence [0..1] : CharacterString
1
1
1
+serviceIdentification
0..1
0..1
+serviceProvider
ServiceIdentification
(from OWS Service Identification)
ServiceProvider
(from OWS Service Provider)
0..1
+operationsMetadata
OperationsMetadata
(from OWS Operations Metadata)
RequestBase {Abstract}
+ service : CharacterString
+ request : CharacterString
+ version : CharacterString

Figure B.1 — OWS Common UML package

B.2 OWS Service Identification package

Figure B.2 shows the OWS Service Identification UML package that models the contents of the ServiceIdentification section of all service metadata documents, as specified in Subclauses 7.4.3 and 7.4.7. In addition to the ServiceIdentification class, this diagram shows the Keywords class used from ISO 19115: Metadata and the Code class used from GML 3. The ServiceIdentification class introduced by this package is further defined by Table 8 in this document.

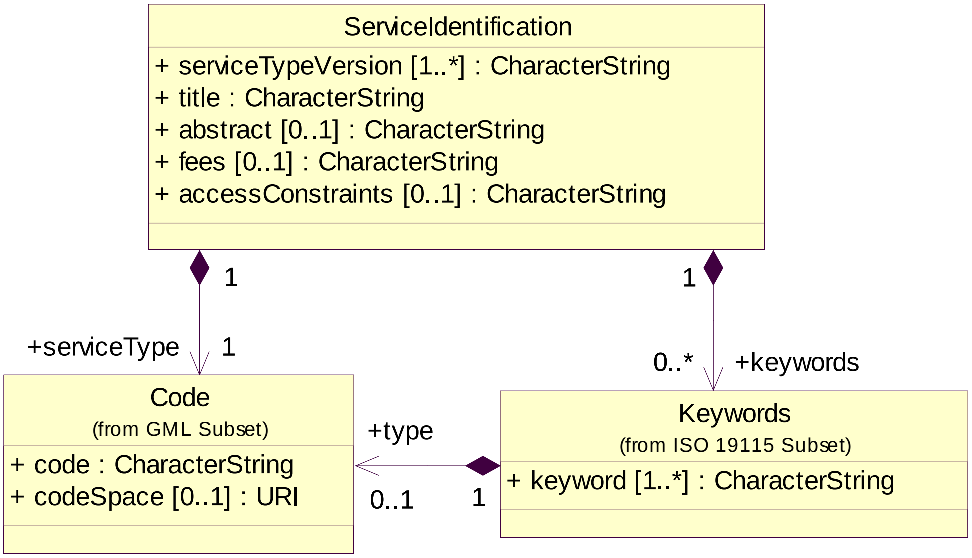
ServiceIdentification
+ serviceTypeVersion [1..*] : CharacterString
+ title : CharacterString
+ abstract [0..1] : CharacterString
+ fees [0..1] : CharacterString
+ accessConstraints [0..1] : CharacterString
1
1
+serviceType
1
0..*
+keywords
Code
(from GML Subset)
+ code : CharacterString
+ codeSpace [0..1] : URI
+type
0..1
1
Keywords
(from ISO 19115 Subset)
+ keyword [1..*] : CharacterString

Figure B.2 — OWS Service Identification UML package

B.3 OWS Service Provider package

Figure B.3 shows the OWS Service Provider UML package that models the contents of the ServiceProvider section of all Service Metadata documents, as specified in Subclauses 7.4.4 and 7.4.9. In addition to the ServiceProvider class, this diagram shows the various classes used from ISO 19115: Metadata. The ServiceProvider class introduced by this package is further defined by Table 9 in this document.

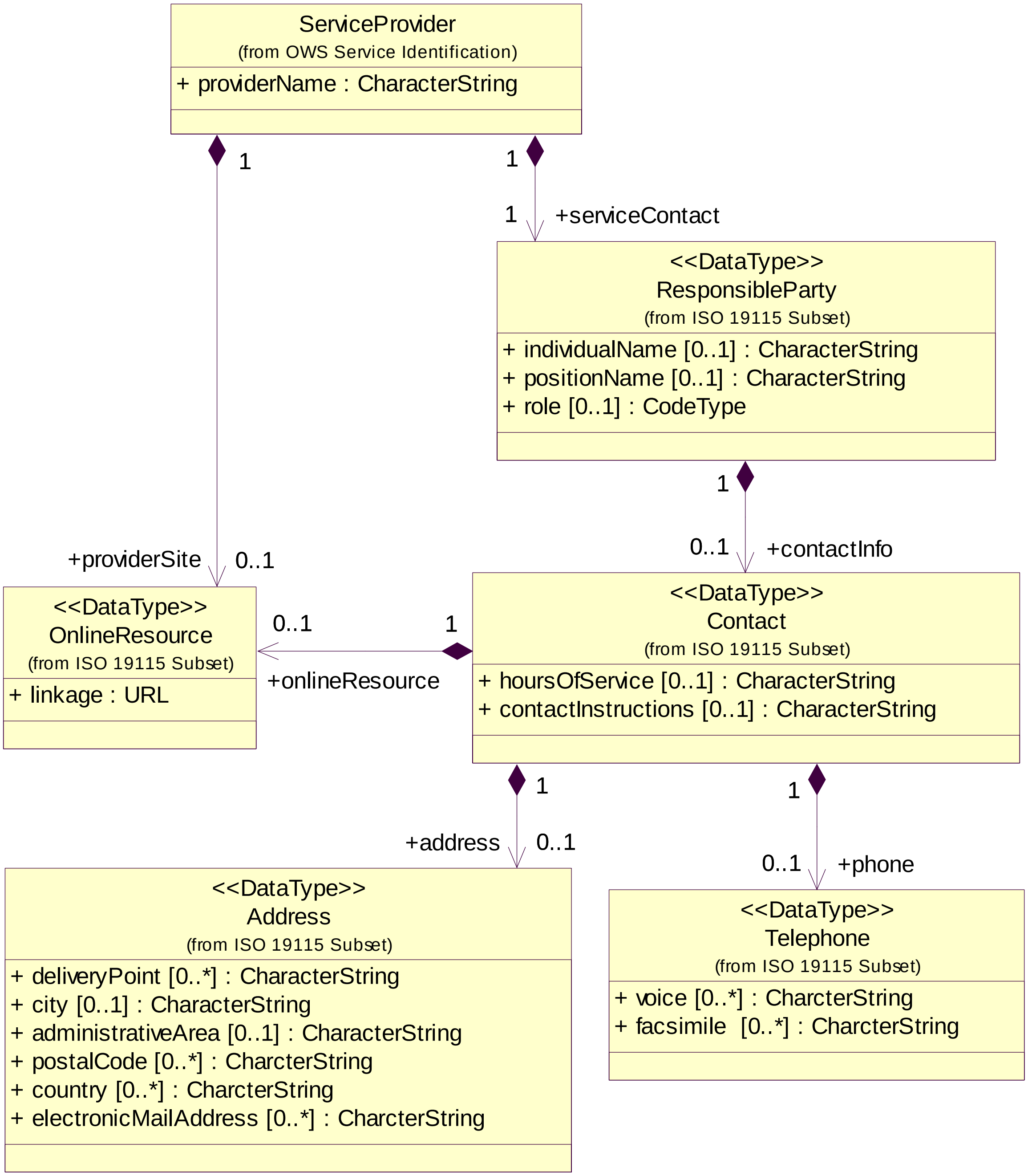
ServiceProvider
(from OWS Service Identification)
+ providerName : CharacterString
1
1
1
+serviceContact
<<DataType>>
ResponsibleParty
(from ISO 19115 Subset)
+ individualName [0..1] : CharacterString
+ positionName [0..1] : CharacterString
+ role [0..1] : CodeType
1
0..1
+contactInfo
+providerSite
0..1
<<DataType>>
OnlineResource
(from ISO 19115 Subset)
+ linkage : URL
0..1
1
+onlineResource
<<DataType>>
Contact
(from ISO 19115 Subset)
+ hoursOfService [0..1] : CharacterString
+ contactInstructions [0..1] : CharacterString
1
1
+address
0..1
0..1
+phone
<<DataType>>
Address
(from ISO 19115 Subset)
+ deliveryPoint [0..*] : CharacterString
+ city [0..1] : CharacterString
+ administrativeArea [0..1] : CharacterString
+ postalCode [0..*] : CharcterString
+ country [0..*] : CharcterString
+ electronicMailAddress [0..*] : CharcterString
<<DataType>>
Telephone
(from ISO 19115 Subset)
+ voice [0..*] : CharcterString
+ facsimile [0..*] : CharcterString

Figure B.3 — OWS Service Provider UML package

B.4 OWS Operations Metadata package

Figure B.4 shows the OWS Operations Metadata UML package that models the contents of the OperationsMetadata section of service metadata documents, as specified in Subclauses 7.4.5 and 7.4.9. In addition to the OWS specific classes, this diagram shows the OnLineResource class used from ISO 19115: Metadata. The Operations Metadata and other classes introduced by this package are further defined by Tables 10-15 in this document.

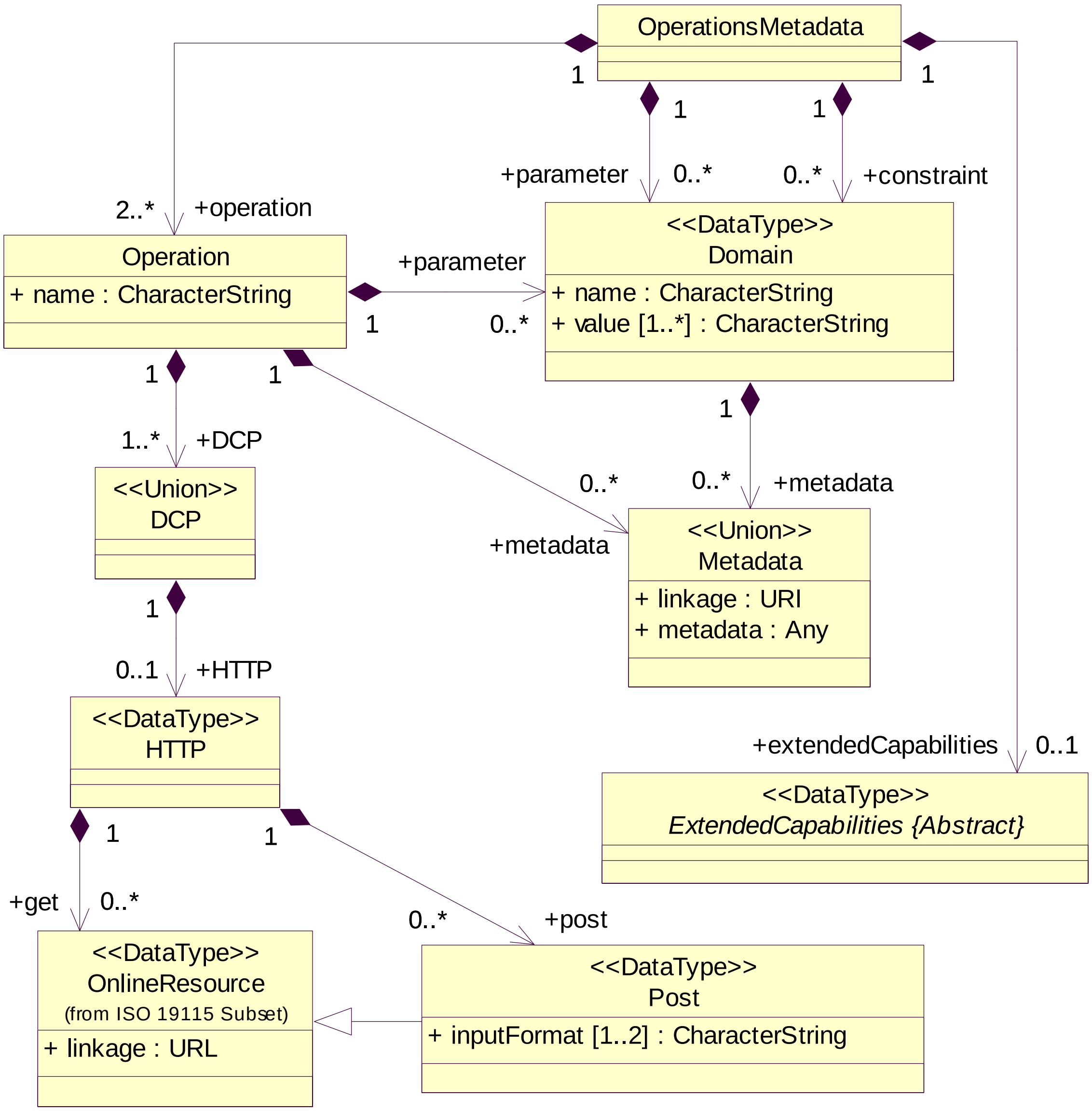

Figure B.4 — OWS Operations Metadata UML package

B.5 OWS Exception Report package

Figure B.5 shows the OWS Exception Report UML package that models the contents of the Exception Reports, as specified in Clause 8. The two classes introduced by this package are further defined by Tables 18 and 19 in this document.

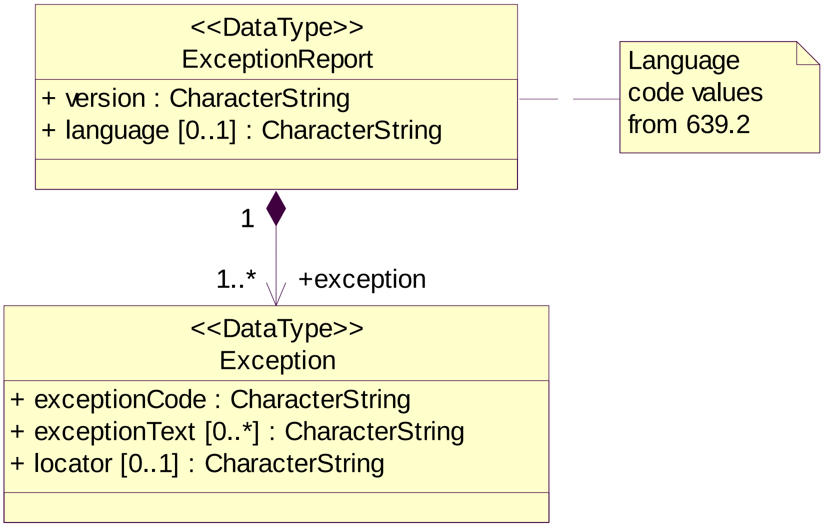
<<DataType>>
ExceptionReport
+ version : CharacterString
+ language [0..1] : CharacterString
Language code values from 639.2
1
1..*
+exception
<<DataType>>
Exception
+ exceptionCode : CharacterString
+ exceptionText [0..*] : CharacterString
+ locator [0..1] : CharacterString

Figure B.5 — OWS Exception Report UML package

B.6 OWS Bounding Box package

Figure B.5 shows the OWS Bounding Box UML package that models the contents of the Bounding Boxes specified in Subclause 10.2. The two classes introduced by this package are further defined by Tables 22 and 23 in this document.

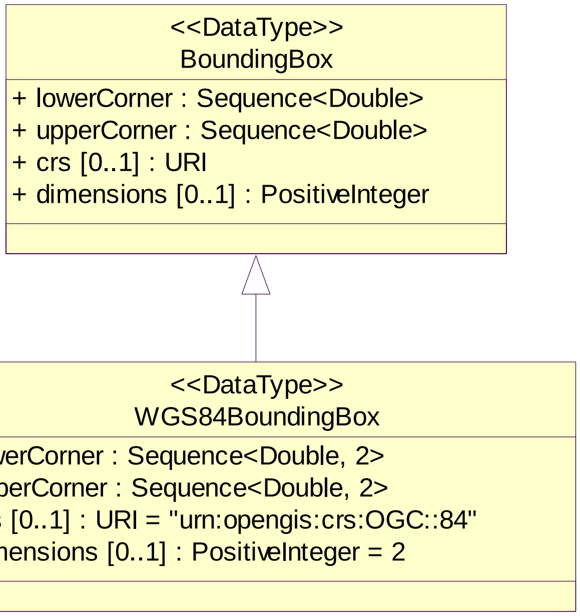
<<DataType>>
BoundingBox
+ lowerCorner : Sequence<Double>
+ upperCorner : Sequence<Double>
+ crs [0..1] : URI
+ dimensions [0..1] : PositiveInteger
<<DataType>>
WGS84BoundingBox
+ lowerCorner : Sequence<Double, 2>
+ upperCorner : Sequence<Double, 2>
+ crs [0..1] : URI = "urn:opengis:crs:OGC::84"
+ dimensions [0..1] : PositiveInteger = 2

Figure B.6 — OWS Bounding Box UML package

Annex C
(informative)

Reasons for parameters

C.1 Introduction

This annex briefly states reasons for deciding to include parameters in the various operation request and response messages specified in this document.

C.2 Reasons for GetCapabilities request parameters

The reasons for deciding to include the parameters listed in the GetCapabilities operation request, in Subclause 7.2.2, are briefly stated in the Table C.1.

Table C.1 — Reasons for GetCapabilities request parameters

Name	Reason
service	The service must be identified in all operation requests because a single endpoint may implement more than one service, and the same operation name may be used by multiple service types.
request	The requested operation must be identified in all operation requests because a single endpoint may implement more than one operation.
version	(Deprecated: The specification version was optional in the GetCapabilities operation request for all OGC Web Services to support client-server version negotiation.)
AcceptVersions	The AcceptVersions is optional in the GetCapabilities operation request for all OWSs, but its use is encouraged to support efficient client-server version negotiation as specified in Subclause 7.3.2. It is optional partly to support backwards compatibility as discussed in Subclause C.11.
Sections	The Sections parameter should be optional in the GetCapabilities operation request for all OWSs to allow clients to request, and servers to respond, with only the needed part(s) of the complete service metadata document. The allowed value of "All" is included to allow a client to request all sections, when that doing so is easier than omitting this parameter. This parameter is significantly more flexible than required by the use cases that have been identified, but is hoped to be cost-effective. See summary of the use cases identified in Subclause C.4.
updateSequence	The updateSequence should be optional in the GetCapabilities operation request for all OWSs to allow clients to request, and servers to respond, with a service metadata document only when it has been updated since the last version returned (see Subclause 7.3.3).
AcceptFormats	The AcceptFormats parameter is included to provide flexibility to allow experimentation and allow other documents to identify allowed alternative format(s). [a]

[a] The AcceptFormats parameter can be optional in the GetCapabilities operation request for all OWSs to allow clients to request different formats for return of service metadata (Capabilities) XML documents. However, implementation should not be required by servers or clients, partially for backwards compatibility. The GetCapabilities operation is unique in that it is the only operation for which the client cannot be expected to know anything about the server. In order to avoid complicated negotiation rounds such as the deprecated version-negotiation mechanism, a GetCapabilities request should contain enough information so that the server can respond to a GetCapabilities request in a way that best satisfies the needs of the client. For this reason, the single "format" parameter that exists in some other operation requests does not suffice. The AcceptFormats parameter allows the client and server to negotiate the best return format in one round. This is similar to the AcceptVersions parameter which has been added to the GetCapabilities request for the same reason. The common first use of this parameter is likely to be by clients that are capable of receiving XML documents in a format which is much smaller than the equivalent text/xml documents. This parameter allows such clients to request the preferred format(s) while still negotiating down to text/xml for servers that do not support the preferred format(s).

The reasons for deciding to NOT include some possible parameters in the GetCapabilities operation request are briefly stated in the Table C.2.

Table C.2 — Reasons against GetCapabilities parameters

Definition	Data type and value	Reason
Identifier of server configuration (data or other)	Character String type, not empty Values are selected by each server	Different server configurations can and should use different URLs [a]
Expression to retrieve any subsection(s) of capabilities document	anyURI or XPATH expression type, not empty (optional) Values are selected by clients	Complexity of implementing extraction of any capabilities document subsection
[a] At the client-server interface, different server configurations are logically separate servers. Implementations of these logically separate servers can share software, and may also share selected data, but such sharing is not explicitly visible to clients.		

C.3 Reasons for service metadata sections

The reasons for deciding to organize the service metadata (or Capabilities) document into the sections listed in the Service metadata document contents, in Subclause 7.4.2, are briefly stated in the Table C.3.

Table C.3 — Reasons for service metadata sections

Name	Reason
ServiceIdentification	Corresponds to and expands the SV_ServiceIdentification class in ISO 19119
ServiceProvider	Corresponds to and expands the SV_ServiceProvider class in ISO 19119
OperationsMetadata	Contains set of Operation elements that each corresponds to and expand the SV_OperationsMetadata class in ISO 19119
Contents	Whenever relevant, contains set of elements that each corresponds to the MD_DataIdentification class in ISO 19119 and 19115

C.4 Reasons for ServiceIdentification parameters

The reasons for deciding to include the parameters and subsections listed in the ServiceIdentification section, in Subclause 7.4.3, are briefly stated in the Table C.4.

Table C.4 — Reasons for ServiceIdentification parameters

Name	Reason
ServiceType (mandatory)	Useful to provide service type name useful for machine-to-machine communication
ServiceTypeVersion (mandatory)	Useful to provide list of server-supported versions.
Title (mandatory)	Useful to provide a server title for display to a human.
Abstract (optional)	Usually useful to provide narrative description of server for display to a human.
Keywords (optional)	Often useful to provide keywords useful for server searching.
Fees (optional)	Usually useful to specify fees, or NONE if no fees.
AccessConstraints (optional)	Usually useful to specify access constraints, or NONE if no access constraints.

Other reasons for deciding to include most of the parameters listed in the ServiceIdentification section were that they either or both:

a) Contain the same information as previously included in both the WMS and WFS Implementation Specifications, in the Service section

b) Are included in the service metadata specified in ISO 19119, in the SV_ServiceIdentification and MD_Identification classes

NOTE 1 ISO 19119 specifies service metadata in Subclauses 7.4.2 and C.2. Almost all this service metadata is for a specific server instance, not for a general service type. This is consistent with the “ServiceIdentification” section of an “OWS Capabilities” document, in which most contents are server specific. Almost all general service metadata is assumed to be available in the Implementation Specification, and to be known by clients.

NOTE 2 Although the revised DIS_(E) version of ISO 19119 no longer lists the association to MD_Keywords in Subclause C.2.2, we assume the association of MD_Identification to MD_Keywords in ISO 19115 can be included when useful.

Table C.5 lists the names of the parameters in the ServiceIdentification section with the names of corresponding Parameters. The centre column lists all the parameter names in the ServiceIdentification section, and the left column lists the corresponding parameter names in the Service section of BOTH WMS and WFS. The right column lists the corresponding UML attribute names in the SV_ServiceIdentification and MD_Identification classes specified in ISO 19119. The right column does NOT list optional UML attributes that have no corresponding parameter in the ServiceIdentification section. All three columns uses dot-separator notation to specify parts of a parent item.

Table C.5 — Corresponding parameter names

WMS and WFS parameter	ServiceIdentification parameter	ISO 19119 UML attribute
Name (mandatory)	ServiceType (mandatory)	SV_ServiceIdentification. serviceType (mandatory)
(none)	ServiceTypeVersion (mandatory)	SV_ServiceIdentification. serviceTypeVersion (optional)
Title (mandatory)	Title (mandatory)	MD_Identification.citation.title (mandatory, mandatory)
(none)	(none)	MD_Identification.citation.date (mandatory, mandatory)
Abstract (optional)	Abstract (optional)	MD_Identification.abstract (mandatory)
Fees (optional)	Fees (optional)	SV_ServiceIdentification. accessProperties.fees (optional, optional)
AccessConstraints (optional)	AccessConstraints (optional)	SV_ServiceIdentification. restrictions.accessConstraints (optional, optional)
KeywordList (optional)	Keywords.Keyword (optional,mandatory)	MD_Identification. keywords.keyword (optional, mandatory)
(none)	Keywords.Type.string (optional, optional, mandatory)	MD_Identification.keywords.type (optional, optional)
(none)	Keywords.Type.codeSpace (optional, optional, optional)	MD_Identification. keywords.thesaurus (optional, optional)

C.5 Reasons for ServiceProvider parameters

The reasons for deciding to include the parameters and subsections listed in the ServiceProvider section, in Subclause 7.4.4, are briefly stated in the Table C.6.

Table C.6 — Reasons for ServiceProvider parameters and subsections

Name	Reason
ProviderName (mandatory)	Useful to allow providing name of service provider at top level.
ProviderSite (optional)	Usually useful to provide reference to provider web site.
ServiceContact (mandatory)	Often useful to provide service provider contact information.

Other reasons for deciding to include most of the parameters and subsections listed in the ServiceIdentification section were that they either or both:

a) Contain the same information as previously included in both the WMS and WFS Implementation Specifications, in the Service section

b) Are included in the service metadata specified in ISO 19119, in the SV_ServiceProvider class

Table C.7 lists the names of the parameters in the ServiceProvider section with the names of corresponding Parameters. The centre column lists all the parameter names in the ServiceProvider section, and the left column lists the corresponding parameter names in the Service section of BOTH WMS and WFS. The right column lists the corresponding UML attribute names in the SV_ServiceProvider class specified in ISO 19119. The right column does NOT list optional UML attributes that have no corresponding parameter in the ServiceIdentification section. All three columns uses dot-separator notation to specify parts of a parent item.

Table C.7 — Corresponding parameter names

WMS and WFS parameter	ServiceProvider parameter	ISO 19119 UML attribute
ContactPersonPrimary. ContactOrganization (optional, optional, mandatory)	ServiceProvider.ProviderName (optional, mandatory)	SV_ServiceProvider.providerName (optional, mandatory) SV_ServiceProvider.serviceContact. organizationName (optional, mandatory, mandatory)
OnlineResource (mandatory)	ServiceProvider.ProviderSite (optional, optional)	(none)
ContactInformation. ContactPersonPrimary. ContactPerson (optional, optional, mandatory)	ServiceProvider. ServiceContact. IndividualName (optional, mandatory, optional)	SV_ServiceProvider.serviceContact. individualName (optional, mandatory, optional)
ContactInformation. ContactPersonPrimary. ContactPosition (optional, optional, mandatory)	ServiceProvider. ServiceContact. PositionName (optional, mandatory, optional)	SV_ServiceProvider.serviceContact. positionName (optional, mandatory, mandatory)
(none)	ServiceProvider. ServiceContact.Role (optional, mandatory, optional)	SV_ServiceProvider.serviceContact. role (optional, mandatory, mandatory)
(none)	ServiceProvider. ServiceContact. ContactInfo.OnlineResource (optional, mandatory, optional, optional)	SV_ServiceProvider.serviceContact. contactInfo.onlineResource (optional, mandatory, optional, optional)
(none)	ServiceProvider.ServiceContact. ContactInfo.HoursOfService (optional, mandatory, optional, optional)	SV_ServiceProvider.serviceContact. contactInfo.hoursOfService (optional, mandatory, optional, optional)
(none)	ServiceProvider ServiceContact. ContactInfo.ContactInstructions (optional, mandatory, optional, optional)	SV_ServiceProvider.serviceContact. contactInfo.contactInstructions (optional, mandatory, optional, optional)
ContactInformation. ContactVoiceTelephone (optional, optional)	ServiceProvider. ServiceContact. ContactInfo.Phone.Voice (optional, mandatory, optional, optional)	SV_ServiceProvider.serviceContact. contactInfo.phone.voice (optional, mandatory, optional, optional, optional)

WMS and WFS parameter	ServiceProvider parameter	ISO 19119 UML attribute
ContactInformation. ContactFacsimileTelephone (optional, optional)	ServiceProvider. ServiceContact. ContactInfo.Phone.Facsimile (optional, mandatory, optional, optional)	SV_ServiceProvider.serviceContact. contactInfo.phone.facsimile (optional, mandatory, optional, optional, optional)
ContactInformation. ContactAddress.AddressType (optional, optional, mandatory)	(none)	(none)
ContactInformation. ContactAddress.Address (optional, optional, mandatory)	ServiceProvider.ServiceContact. Address.DeliveryPoint (optional, mandatory, optional, optional)	SV_ServiceProvider.serviceContact. contactInfo.address.deliveryPoint (optional, mandatory, optional, optional, optional)
ContactInformation. ContactAddress.City (optional, optional, mandatory)	ServiceProvider.ServiceContact. Address.City (optional, mandatory, optional, optional)	SV_ServiceProvider.serviceContact. contactInfo.address.city (optional, mandatory, optional, optional, optional)
ContactInformation. ContactAddress. StateOrProvince (optional, optional, mandatory)	ServiceProvider.ServiceContact. Address.AdministrativeArea (optional, mandatory, optional, optional)	SV_ServiceProvider.serviceContact. contactInfo.address. administrativeArea (optional, mandatory, optional, optional, optional)
ContactInformation. ContactAddress.PostCode (optional, optional, mandatory)	ServiceProvider.ServiceContact. Address.PostalCode (optional, mandatory, optional, optional)	SV_ServiceProvider.serviceContact. contactInfo.address.postalcode (optional, mandatory, optional, optional, optional)
ContactInformation. Contact.Address.Country (optional, optional, mandatory)	ServiceProvider.ServiceContact. Address.Country (optional, mandatory, optional, optional)	SV_ServiceProvider.serviceContact. contactInfo.address.country (optional, mandatory, optional, optional, optional)
ContactInformation. Contact.ElectronicMailAddress (optional, optional)	ServiceProvider. ServiceContact. ElectronicMailAddress (optional, mandatory, optional, optional)	SV_ServiceProvider.serviceContact. contactInfo.address. electronicMailAddress (optional, mandatory, optional, optional, optional)

C.6 Reasons for OperationsMetadata parameters

The reasons for deciding to include the parameters and subsections listed in the OperationsMetadata section, in Subclause 7.4.5, are briefly stated in the Table C.8.

Table C.8 — Reasons for OperationsMetadata parameters and subsections

Name	Reason
Operation.name (mandatory)	Required to identify operation request.
Operation.DCP.HTTP.Get (mandatory, optional)	Required to identify Get connect point of HTTP Distributed Computing Platform.
Operation.DCP.HTTP.Post.URL (mandatory, optional)	Required to identify Post connect point URL of HTTP Distributed Computing Platform.
Operation.DCP.HTTP.Post .InputFormat (mandatory, optional, optional)	Required to identify Post format(s) of HTTP Distributed Computing Platform, either XML or KVP encoded. Optional so can be omitted for most common case, and for backwards compatability.
Operation.Parameter.name (optional, mandatory)	Required to identify constrained parameter.
Operation.Parameter.Value (optional, mandatory)	Required to specify allowed value of constrained parameter.
Operation.Parameter.Metadata (optional, mandatory)	Sometimes useful for constrained parameter information.
Operation.Metadata (optional)	Sometimes useful for specific operation information, sometimes multiple metadata sets useful.
OperationsMetadata. Parameter.name (optional, mandatory)	Required to identify constrained parameter.
OperationsMetadata. Parameter.Value (optional, mandatory)	Required to specify allowed value of constrained parameter.
OperationsMetadata. Parameter.Metadata (optional, mandatory)	Sometimes useful for constrained parameter information, sometimes multiple metadata useful.
Constraint.name (optional, mandatory)	Required to identify constrained quantity.
Constraint.Value (optional, mandatory)	Required to specify allowed value of constrained quantity.
Constraint.Metadata (optional, mandatory)	Sometimes useful for constrained quantity information.
ExtendedCapabilities (optional)	Sometimes useful for server-specific abilities.

Other reasons for deciding to include many of the parameters and subsections listed in the OperationsMetadata section were that they either or both:

a) Contain the same information as previously included in both the WMS and WFS Implementation Specifications, in the Capability section

d) Are included in the service metadata specified in ISO 19119, in the SV_OperationMetadata class

C.7 Reasons for all operations except GetCapabilities minimum parameters

The reasons for deciding to include the minimum parameters listed for all operation request and response messages except GetCapabilities, in Clause 9, are briefly stated in the Table C.9.

Table C.9 — Reasons for parameters in all operations except GetCapabilities

Name	Reason
service	The service must be identified in all operation requests because a single endpoint may implement more than one service, and the same operation name may be used by multiple service types.
request	The requested operation must be identified in all operation requests because a single endpoint may implement more than one operation.
version	The specification version must be identified in all operation requests except GetCapabilities because a single server may support more than one version of a specification, and thus needs to know the specific version of the operation being requested.

C.8 Reasons for Exception Report parameters

The reasons for deciding to include the parameters listed for Exception Reports, in Clause 8, are briefly stated in the Table C.10.

Table C.10 — Reasons for Exception Report parameters

Name	Reason
ExceptionText	Encourage inclusion of text providing more information about exception
exceptionCode	Require inclusion of one of a set of specified exception type identifiers for each exception reported to clients
locator	Often useful to provide an indicator of the location in the client's operation request where this exception was encountered
version	Require inclusion of “version” of exception report XML Schema
language	Allow and encourage identification of the language used for ExceptionText values

The reasons for deciding to NOT include some possible parameters in Exception Reports are briefly stated in the Table C.11.

Table C.11 — Reasons against Exception Report parameters

Definition	Data type and value	Reason
Reference to on-line resource used by this operation request	URL type Value is URL that received this operation request	We cannot now identify a use for this parameter [a]
Service type identifier	Character String type, not empty Value is OWS type abbreviation (e.g., "WMS", "WFS")	We believe the software that sends an operation request will always receive any corresponding Exception Report [a]
a Future handling of chained services might require adding parameters such as these.		

C.9 Use cases for Sections parameter

The following uses cases have been identified for the Sections parameter, in a GetCapabilities operation request:

a) A client does NOT need the Contents part of the service metadata document, but needs one or more other parts. This need will occur when either:

   1) The client expects that the Contents part of the service metadata document is large, and wants to first check one or more other parts to see if the client can and should use this server.

   2) The client has access through another service (e.g., a catalog service) to all the client-needed data contained in the Contents part of the service metadata document.

NOTE 1 If all parts except the Contents part of the service metadata document are relatively small, it would be acceptable to return all parts except the Contents part of the service metadata document even if only one other part is needed.

b) A client needs only the Contents part of the service metadata document. This need will occur when either:

   1) The client has already retrieved the rest of the service metadata document, and now needs the Contents part.

   2) The client has already retrieved the complete service metadata document, and now needs to check for any update (which is most likely to affect the Contents part).

NOTE 2 If all parts except the Contents part of the service metadata document are relatively small, it would be acceptable to return all parts of the service metadata document even if only the Contents part is needed.

c) A client needs all parts of the service metadata document. This need will occur when both:

   1) The client has not yet retrieved any part of the service metadata document, and now needs several parts.

   2) The client does not have access through another service (e.g., a catalog service) to the client-needed data contained in the Contents part of the service metadata document.

C.10 Requirements for exception reports

The assumed requirements for exception reports include:

a) Allow one exception report to report multiple errors

b) Allow exception report to report a hierarchy of error indicators for one basic error

c) Require exception reports to report one of a set of specified error types for each error

d) Strongly encourage exception reports to also report ad-hoc information about each error

e) Allow ad-hoc error information to be in various languages

C.11 Version negotiation backward compatibility

The new version negotiation process, defined in Subclause 7.3.2, was designed to be compatible with the old-style version negotiation defined in earlier versions of the various OWS specifications. That old-style version negotiation used the optional "version" parameter in a GetCapabilities request.

The old-style version negotiation process stated that if the "version" parameter is missing, then a service metadata document compliant to the highest-supported version shall be returned. Therefore, if a new client sends a GetCapabilities request containing an AcceptVersions parameter to an old server that does not recognize it, the server will return a service metadata document compliant to the highest version that it supports. The client will either recognize this version, in which case version negotiation has been successful, or it does not. In the situation where the client sees a service metadata document for a version that it does not support, the client may optionally revert back to the old-style version negotiation mechanism to complete the negotiation.

A server may also optionally implement the old-style version negotiation mechanism so that old clients that send GetCapabilities requests containing a "version" parameter can be served. If both a "version" and an AcceptVersions parameter exist in a GetCapabilities request, the server shall ignore the "version" parameter.

a) The old-style version negotiation process using the GetCapabilities operation is as follows: The client initially makes a GetCapabilities operation request identifying the latest version it supports, and then:The server responds to GetCapabilities operation requests:

   1) If no version number is specified in the request, the server shall respond with the highest version it supports.

   2) If the version number specified in the request is supported by the server, the server shall respond with that version.

   3) If the version number specified in the request is lower than the lowest version supported by the server, the server shall respond with the lowest version that it supports.
   4) If the version number specified in the request is higher than the lowest version supported by the server, the server shall respond with the highest version it supports that is lower than the requested version.

b) The client reacts to GetCapabilities operation responses:
   1) If the response version is supported by the client, then version negotiation is complete and successful.
   2) Otherwise, if the response version is lower than the requested version, and if the client supports some version lower than the response version, the client shall make another request with the highest version it supports that is lower than the response version, and this process is repeated.
   3) Otherwise (if the response version is higher than the request version, or if the response version is lower than the request version and the client supports no version lower the response version), then version negotiation was unsuccessful.

C.12 Bounding box requirements

The assumed requirements for bounding boxes included:

a) Define bounding boxes that can be used for 1D through 4D or more dimensions of spatial and/or temporal extents.

b) Not specify any specialized time range, to be used separately or as an extension to a spatial bounding box. If needed by a specific OWS, allow a spatial bounding box to be extended by a time range whose format is specified.

c) Allow bounding boxes to be repeated whenever useful. A specific OWS can use repeated bounding boxes for any use with any meaning that it specifies. One expected use would mean the union of the areas defined by multiple listed bounding boxes.

d) For ellipsoidal coordinate systems, allow one bounding box to extend completely around the ellipsoid. Similarly, for any continuous circular angle coordinate axis (in spherical, polar, and cylindrical coordinate systems), allow one bounding box to extend around the complete circle.

e) For any coordinate axis containing a value discontinuity, allow the bounding box to extend across that discontinuity and still be the minimum size box surrounding the data. For example for geographic CRSs, allow the bounding box Longitude to extend across the +/– 180 degrees meridian but not extend completely around the spheroid. Similarly for spherical, polar, and cylindrical coordinate systems, allow the bounding box to extend across the discontinuity in the (otherwise continuous) angle.

f) In the OWS Common Implementation Specification, do not allow use of any unspecified geographic CRS, with the Greenwich prime meridian assumed but not documented. (Not like the approximate EX_GeographicBoundingBox in ISO 19115.)

g) Specify only one base spatial-temporal bounding box data structure.

h) Also specify one specialized spatial bounding box, for 2D horizontal geographic coordinates in the WGS 84 CRS with Longitude and Latitude specified in (decimal) degrees.

i) Specify both XML and KVP encodings of the one base spatial-temporal bounding box data structure, and of the one specialized 2D geographic bounding box.

j) Avoid the need to directly use GML 3 schemas, by duplicating needed XML Schema fragments as needed.

C.13 Minimum bounding boxes

The bounding box contents defined in Subclause 10.2 will not always specify the MINIMUM rectangular BOUNDING region, if the referenced CRS uses an Ellipsoidal, Spherical, Polar, or Cylindrical coordinate system. Specifically, this box will not specify the minimum rectangular bounding region surrounding a geometry whose set of points span the value discontinuity in an angular coordinate axis. That geometry could lie within a small region on the surface of the ellipsoid or sphere. Such axes include the Longitude of Ellipsoidal and Spherical coordinate systems, as specified in OGC Topic 2.

There are a variety of possible approaches to allowing specification of the minimum rectangular bounding region when the referenced CRS uses an Ellipsoidal, Spherical, Polar, or Cylindrical coordinate system. The possible approaches include:

a) Use a general bounding box CRS with angular axes such that the minimum bounding box does not need to cross each axis value discontinuity. For example, use a geographic CRS with its prime meridian near the center of the needed minimum bounding box. Such a CRS can be used, and should be used when practical. However, use of such a CRS constrains the CRS used.

b) For a circular coordinate, specify that the LowerCorner shall define the box edge furthest toward decreasing values, and the UpperCorner shall define the box edge furthest toward larger values. For longitude, the LowerCorner longitude would define the West-most box edge, and the UpperCorner longitude would define the East-most box edge. (The LowerCorner would no longer always use the minimum value, and the UpperCorner would no longer always use the maximum value. The value at the LowerCorner can be greater than at the UpperCorner when this bounding box crossed the value discontinuity.)

c) Allow a circular coordinate value to lie outside the normal value range, so this value can be the minimum or maximum and also define a bounding box that crosses the value discontinuity. For example, allow the LowerCorner longitude to range from minus -540 to +180 degrees (allowing a bounding box from -538 degrees on the West to +179 degrees on the East).

d) Use two or more bounding boxes, one on each side of the each axis value discontinuity within the desired minimum bounding box. The specified minimum rectangular bounding region would then be the union of those multiple bounding boxes. For example, use two bounding boxes with one on each side of the +/– 180 degrees meridian of a geographic CRS.

e) Use a bounding box that includes all allowed circular coordinate axis values, although this box will usually be much wider than the minimum. For example, use a bounding box that includes all longitude values, interpreted as meaning this box is continuous around the earth (when the data may extend only from +179 degrees on the West to -179 degrees on the East).

f) Add an optional "Offset" (or "Center") position to the general Bounding Box, to be added to the LowerCorner and UpperCorner positions. Include this Offset position when needed to specify a minimum bounding box that crosses the value discontinuity in one or more axes. This would allow the Offset, LowerCorner, and UpperCorner positions to all lie within the normal axes range limits. When needed, this Offset position could be set to the (approximate) center of the minimum bounding box.

g) Redefine the general Bounding Box to contain mandatory "Center" and "Offset" positions. This Offset would always be added to and subtracted from this Center to obtain the lower corner and upper corner positions.

EDITOR'S NOTE As stated, alternative a) can always be used by proper selection of the CRS, and should be used when practical. In most other alternatives listed above, server and client implementations would be more difficult than desired. Abstract Specification Topic 11 (ISO 19115) uses a form of alternative b), and Doug Nebert has stated that FGDC implementations use alternative b). Roel Nicolai has expressed a preference for alternatives c) or b). The (not-strong) recommendation is thus to use alternative b).

C.14 CRS reference requirements

C.14.1 Introduction

This subclause summarizes many of the requirements considered when specifying how to reference CRSs in Subclause 10.3. Much of this material is also applicable to CRS components and to Coordinate Operations and their components. This material builds on the abstract specifications of CRS objects and components in [OGC Topic 2]. That topic also defines Coordinate Operations which specify how to transform coordinates between two different CRSs. There are two primary subtypes of Coordinate Operations, namely Transformations and Conversions.

C.14.2 CRS identifiers and definitions

Each specific CRS, and each component thereof, usually has multiple identifiers. The definition of a CRS includes one or more identifications of that CRS, as specified in Subclause 8.2 of [OGC Topic 2]. Furthermore, none of those identifications is encoded in the form of URI, and a URI encoding of identification is rarely unique. So many different URI encodings of a CRS identifier are possible for the same CRS.

For interoperability among multiple servers and multiple clients, it is important that they all use the same CRS identifier for the same CRS. This commonality of identifiers should be partially achieved by suitable restricting the allowed forms and formats of identifiers.

This commonality of identifiers should also be partially achieved by OGC Implementation Specifications defining some standard identifiers and parts of compound identifiers.

If software needs to transform coordinate positions into an identified CRS from another CRS, it often needs only a recognizable identifier of the desired CRS, not its complete definition. A CRS definition does not include enough information to transform coordinate positions from or to any other CRS, except for the Projected and Derived types of CRSs. For those types of CRSs, the CRS definition includes or identifies the one coordinate Conversion needed to transform coordinate positions to or from a reference “base” CRS.

For all other types of CRSs, one or more coordinate Transformation specifications are needed to transform coordinate positions between two CRSs. Each coordinate Transformation specification identifies its’ source and target CRS, but does not depend on any other information in the definitions of those two CRSs. No CRS definition includes or references any coordinate Transformation specification.

C.14.3 Use cases for CRS identifiers

The following use cases have been identified for coordinate reference system (CRS) identification parameters in OWS operation requests and responses:

a) A client needs to identify a CRS to a server in an operation request, where that CRS is already “known” to that server. This need can occur when the server already knows about that CRS because either:
   1) That server identifies that CRS in its service metadata (or Capabilities) document, usually in the Contents section of that document
   2) The specific Implementation Specification implemented by that server sufficiently defines that CRS, indicating that (all or many) server implementations should be able to handle that CRS
   3) That server uses that CRS identifier to (try to) find or select a coordinate Transformation or Conversion which handles that CRS (not depending on this client)
   4) That server identifies that CRS definition authority in its service metadata (or Capabilities document), indicating that it can handle (all or many) of the CRSs defined by that authority (possible future server ability)

b) A server needs to identify a CRS to a client in an operation response, where the definition of that CRS is NOT needed by that client. This need can occur when the client will:
   1) Return that CRS identification to that server, in an operation request
   2) Display that CRS identification to a person who is likely to recognize that CRS identifier
   3) Use that CRS identifier to find or select a coordinate Transformation or Conversion which handles that CRS (not using this server)

    4) Use the CRS definition provided in the specific Implementation Specification implemented by that client, where that Implementation Specification indicates that many client implementations should be able to handle that CRS

c) A server needs to identify a CRS to a client in an operation response, where the definition of that CRS also IS needed by that client. In this case, the CRS definition might be included in the same operation response, in a different operation response, or be electronically available to the client from another source known to the server. This need can occur when the client will:

    1) Display that CRS definition to a person who may want that information
    2) Use the coordinate Conversion included (or referenced) in that CRS definition, to transform coordinates to or from the identified base CRS

d) A client needs to identify a CRS to a server in an operation request, where the definition of that CRS also IS needed by that server. In this case, the CRS definition might be included in the same operation request, in a previous operation request, or be electronically available to the server from another source known to the client. This need can occur when that server will:

    1) Use the coordinate Conversion included, implied, or referenced in that CRS definition, to transform coordinates to or from the identified base CRS

NOTE The “AUTO” CRSs currently specified in Annex B of the WMS specification [OGC WMS] are of this nature. These CRSs are Projected CRSs that use one of five defined coordinate Conversions, each with identified parameter values left unspecified. The reference to this AUTO CRS then includes specific values for that one or two parameters.

In items c) and d) listed above, the other source electronically available (to the client and/or server) could be a catalogue or registry of such CRS definitions, or a set of web pages containing these CRS definitions (probably encoded in XML). At present, no fairly comprehensive catalogue or set of web pages containing CRS definitions is known to be publicly available. However, more than one partially-overlapping CRS definition catalogue or set of web pages are expected to become publicly available over the next few years.

C.14.4 CRS identifier requirements

A CRS identifier shall be allowed to be of xsd:anyURI type, as that type is specified in XML Schema. This anyURI type is currently used for CRS references in many places, including in GML (including in Clause 12 of GML 3.1). Furthermore, the anyURI type allows several alternative types of identifiers, including URLs and multiple types of URNs. However, other types of CRS identifiers could also be allowed if a URI is considered to be too cumbersome.

The assumed requirements on the allowed forms of CRS references included:

a) Allow referencing CRSs defined by multiple categories of definition authorities, including:
    1) Allow referencing CRSs completely defined by the OGC and by other organizations recognized by the OGC, such as the EPSG
    2) Allow referencing CRSs whose definitions may be known only by a specific server or client, when the complete CRS definition can be electronically obtained if needed by the recipient of the CRS identifier
    3) Allow referencing CRSs that may not be well-defined, at least when the CRS definition may not be needed by the recipient of the CRS identifier
    4) Allow referencing Projected and Derived CRSs that are not-quite-completely defined by the OGC, or by another organization recognized by the OGC, where the CRS reference is augmented with the missing values of one or a few parameters

NOTE A Projected or Derived CRS is largely specified by an identified coordinate Conversion from an identified base CRS. In a not-quite-completely defined CRS, the identified Conversion may be not-quite-completely specified, in that the values of one or more of the required operation parameters are not (yet) specified. The reference to this CRS could then be augmented with the missing parameter values.

b) Allow referencing CRSs whose definitions have multiple types of availability, including:
    1) Allow referencing definitions electronically available over the Internet/Intranet, at least when those definitions are XML encoded using Clause 12 of GML 3.1
    2) Allow referencing definitions contained in an OGC Implementation Specification, and not electronically available
    3) Allow referencing definitions not electronically available, at least when the definition is adequately specified by a known definition authority

c) Allow referencing (anonymous) CRSs that combine two or three identified components which are each previously defined, including:
    1) Allow referencing not-well-known Compound CRSs that combine two or three previously defined CRSs
    2) Allow referencing not-well-known CRSs that combine a previously defined Coordinate System with a previously defined Datum
    3) Allow referencing not-well-known Projected and Derived CRSs that combine an identified Coordinate System with an identified coordinate Conversion from an identified CRS, when all three are previously defined

C.14.5 CRS definition locations

The definition of a CRS-related object identified by an anyURI must be recorded somewhere outside that URI, and referenced by that URI. Outside that URI might be either:

a) Outside the XML document where it is referenced, such as in a:
    1) Dictionary of definitions available electronically from a specified URL. In this case, an anyURI attribute can reference the dictionary, and the specific item in that dictionary.
    2) Dictionary of definitions not available electronically. However, an anyURI attribute can still reference such a dictionary, and the specific item in that dictionary.
    3) Catalogue service which registers definitions, probably available electronically. In this case, an anyURI attribute can reference the service, and also the specific item available from that service.
b) Inside the XML document where it is referenced, such as in a:
    1) Dictionary of one or more CRS-related definitions stored as metadata in some XML element in that XML document.
    2) Definition included in some XML element specified by an Application Schema based on the GML and/or CRS schemas.

In all cases listed above, the referenced CRS definition could be XML encoded. For example, a XML encoded dictionary of CRS definitions could be similar to the units Dictionary example given in Subclause E.7 of OGC document 03-010r9. When the CRS definition is stored outside the XML document where it is referenced (see item a) above), the remote CRS definition does not need to be encoded in XML.

In some cases, a CRS definition will be well-known to all clients and servers that must interoperate. In such cases, encoding of these well-known definitions is not necessary inside the same XML document and should be avoided when practical. For such well-known definitions, the primary alternatives are storing those definitions either:

a) In a dictionary of definitions not available electronically. For example, such a dictionary could be in a published document, such as an OGC document.
b) In a dictionary of definitions available electronically, often from a specified URL.
c) In a service which registers definitions, probably available electronically.

In other cases, a CRS definition will NOT be well-known to ALL clients and servers that must interoperate. In those cases, the needed definitions might be XML encoded in the same XML document, either in a:

a) Logically appropriate place defined by an Application Schema
b) Metadata element encoded within an XML element that includes all the references to those object definitions, such as the outer-most element of the XML document

C.14.6 URL use by servers

Wherever a specific OWS Implementation Specification allows a URL value to be used for an anyURI in an operation request, it shall specify how servers shall use those URLs in order to be considered compliant with that specification. There are at least three alternative approaches to compliant server uses of such transferred URLs:

a) Optional URL use: Server software is NOT required to automatically access and use data from provided URLs; some or all of this data can be coded into the software. However, (more intelligent) software is allowed to automatically access and use data from (some) provided URLs.

b) Required URL use: Server software IS required to automatically access and use data from provided URLs, if they need to use the referenced data. That is, none of this data can be coded into the software.

c) Selective server URL use: Each server implementation is allowed to specify categories of data that it will automatically access and use from provided URLs, if it needs to use the referenced data. For all other URL referenced data, that server is NOT required to automatically access and use data from that URL; some or all of the other data could be coded into the software.

NOTE When a client uses one or more servers that automatically access and use certain data from provided URLs, that client can provide a URL for the data it wants the server to use. That URL could reference a data server somehow associated with that client, and not necessarily available to other clients.

Bibliography

[1] ISO/IEC 15938-1 - Information Technology - Multimedia Content Description Interface - Part 1: Systems

[2] ISO/TS 19103, Geographic information — Conceptual schema language

[3] OGC 00-014r1, Guidelines for Successful OGC Interface Specifications,

[4] OGC 02-058, Web Feature Service Implementation Specification, v1.0.0

[5] OGC 02-61r2, Web Coordinate Transformation Service Implementation Specification, draft

[6] OGC 03-002r8, Binary-XML Encoding Specification, version 0.0.8

[7] OGC 03-065r6, Web Coverage Service Implementation Specification, v1.0

[8] OGC 03-109r1, Geographic information — Web Map Service interface, v1.3.0