OpenGIS Consortium Discussion Paper #01-022r1

"Basic Services Model Draft Candidate Implementation Specification 0.0.8"

(WWW Mapping Special Interest Group)

This paper presents a discussion of technology issues considered in a Special Interest Group of the Open GIS Consortium Technical Committee. The content of this paper is presented to create discussion in the geospatial information industry on this topic; the content of this paper is not to be considered an adopted specification of any kind. This paper does not represent the official position of the Open GIS Consortium nor of the OGC Technical Committee.

TITLE:	**Basic Services Model Draft Candidate Implementation Specification**	
VERSION:	**0.0.8**	
EDITOR:	**Name:**	**Jeff de La Beaujardiere**
	Address:	**NASA Digital Earth Office**
		Goddard Space Flight Center , Code 933
		Greenbelt MD 20771 USA
	Phone:	**+1 301 286 1569**
	FAX:	**+1 301 286 1777**
	Email:	**delabeau@iniki.gsfc.nasa.gov**
DATE:	**2001-03-05**	
CATEGORY:	**OGC Discussion Paper #01-022r1**	

CONTENTS

1 PREFACE

This document is a partial description of the accomplishments of the Open GIS Consortium Web Mapping Testbed, phase 2 (WMT2). The authors are indebted to those organizations, listed in Section 8, that sponsored and/or participated in WMT2.

1.1 Submitting Organization

This draft implementation specification has been submitted to the OGC as a Discussion Paper by the following organization:

NASA Digital Earth Office

1.2 Submission Contact Points

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

Editor

Jeff de La Beaujardiere
NASA Digital Earth Office
Goddard Space Flight Center, Code 933
Greenbelt MD 20771 USA
+1 301 286 1569
delabeau@iniki.gsfc.nasa.gov

Contributors

Allan Doyle, International Interfaces	adoyle@intl-interfaces.com
John Evans, GST/NASA (formerly MIT)	john.evans@gsfc.nasa.gov
George Percivall, GST/NASA	percivall@gsfc.nasa.gov
Rob Atkinson, Social Change Online	rob@socialchange.net.au

1.3 Document Conventions

This document contains sections that are "normative," meaning they contain the formal specification against which conformance testing can be done. Other sections are "informative," meaning they serve as explanation and background, or "experimental," meaning they describe ideas that were discussed without reaching a satisfactory conclusion or working test implementations. Normative and Experimental sections are explicitly labeled as such.

Experimental sections are shown on a grey background, like so.

In the sections labeled as normative, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [15].

1.4 Revision History

0.0.1 WMT2 development version, 2000 October 4

0.0.4 WMT2 development version, 2000 November 3

0.0.5 WMT2 development version, 2000 November 23

0.0.6 WMT2 development version, 2000 December 8

0.0.7 OGC Discussion Paper #01-022, 2001 January 29

0.0.8 OGC Discussion Paper #01-022r1, 2001 March 6

1.5 Changes To The OpenGIS? Abstract Specification

Pending revisions to Topic 12 of the Abstract Specification are relevant to this specification.

2 INTRODUCTION

2.1 Motivation and Background

The OpenGIS Consortium (OGC) Web Mapping Testbed phase 1 (WMT1) resulted in two documents: "OpenGIS Web Map Server Interface Implementation Specification 1.0.0" (WMS1), and "OpenGIS Recommendation - Geography Markup Language 1.0.0" (GML). These documents were the result of a consensus development process by WMT1 sponsors and participants and were approved by the full OGC membership. GML is an Extensible Markup Language (XML)[9] encoding of geographic features, and is described elsewhere. The WMS 1.0 specification was an important first step towards interoperable access to georeferenced information, but it does have a number of acknowledged limitations. The goal of OGC Interoperability Program 2000 (IP2000), comprising Web Mapping Testbed phase 2 (WMT2) and the Geospatial Fusion Services (GFS) testbed, was to further augment interoperability by enhancing the WMS spec and by creating new standards for Web Coverage Server, Web Feature Server, GeoParser, GeoCoder, GeoLinks, and service Catalogs.

These services--collectively called OGC Web Services--are each the subject of detailed specifications. They are described conceptually in the next section as background material. The specs all differ in their purposes and details, but share a number of common elements. This document describes the basic service framework of OWS and specifies those common elements. In the longer term, this Basic Services Model should evolve into a General Services Model with a better-defined service framework that builds on emerging work on service discovery, description and invocation in the information technology community. The Basic Services Model is an implementation of the ISO TC211 services architecture as found in ISO 19119 Geographic Information – Services.

2.2 Terminology

Terms relating to the abstract definition of geospatial interoperability are: service and operation.

? ? **Service**: A collection of operations, accessible through an interface, that allows a user to evoke a behavior of value to the user (definition from ISO 19119).

? ? **Operation**: specification of an interaction that can be requested from an object to effect behavior (definition from ISO 19119).

Terms relating to the implementation of geospatial interoperability are: Interface and Component.

? ? **Interface:** an implementation of operations including the syntax of the interaction for a given distributed computing technology (definition from ISO 19119).

? ? **Component:** a physical, replaceable part of a system that packages implementation and conforms to and provides the realization of a set of interfaces. Component is synonymous with Server.

(Note that with these definitions, the terms Interface and Server need to be changed in many places in the WMT documentation.)

2.3 Document Organization

This remainder of this document is organized as follows:

Section 3 BACKGROUND (informative)
Provides historical background and technical context for the specifications.

Section 4 ARCHITECTURE (informative)
Describes the OGC Web Services architecture and relates the components to each other.

Section 5 GENERAL RULES (normative)
Specifies the general rules applicable to all or most interfaces of a Web Map, Coverage or Feature Server. This is the primary normative part of this document.

Section 6 THE GETCAPABILITIES OPERATION (normative)
Specifies the request and response rules for obtaining service-level metadata describing an OGC Web Service.

Section 7 REFERENCES (informative)

Section 8 WMT2 PARTICIPANTS (informative)

Appendix A: XML DTDs (normative)

Appendix B: Formatting Dates And Times (normative)

Appendix C: Describing Multi-Dimensional Data (normative)

Appendix D: Multi-Dimensional Coordinate Reference Systems (informative)

4 ARCHITECTURE

4.1 Overview

Three principal types of georeferenced information access services have been defined by WMT2: Web Map Server (WMS), Web Coverage Server (WCS), and Web Feature Server (WFS). In addition, the Geospatial Fusion Services (GFS) Testbed defined services that return spatially referenced results: GeoParser, GeoCoder, and GeoLinker. Collectively, such services are referred to as (OWS). Figure 4-1 is an architecture diagram showing conceptually how OGC Web Services components are related, and naming some (not all) of the interfaces between them. The subject of this document and its companion volumes is the interfaces. The internal details of the components are irrelevant; vendors may build them in any manner that correctly implements the required interfaces.

The WMT2 architecture implements the “User defined chaining” architecture pattern defined in ISO 19119. This pattern is also called transparent chaining as the user is highly aware of and controls the individual services. The transparent pattern depends upon the user to select services through interaction with a service

catalog or registry. (Future OGC Web Services may implement the workflow-managed (or translucent) chaining pattern the in which the Human user invokes a Workflow Management service that controls the chain and the user is aware of the individual services.)

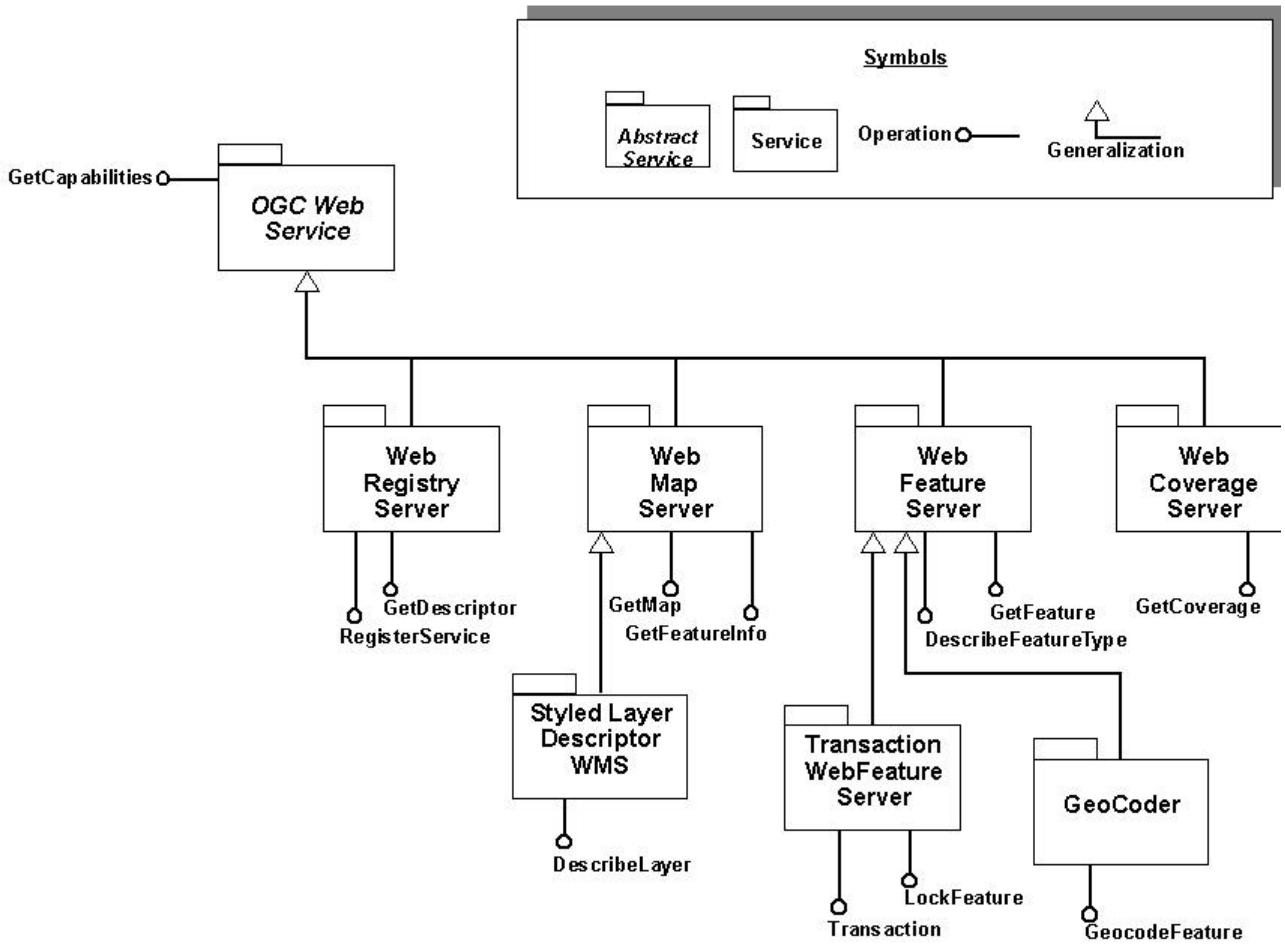

Figure 4.1 - OGC Web Services Architecture diagram.

Each service is described briefly in the following subsections. These descriptions are merely informational; refer to the individual specification documents for complete and normative detail. Note that the only operation common to all services is GetCapabilities. GetCapabilities is specified normatively in this document (Section 6). In brief, that operation allows a Client to request Extensible Markup Language (XML)[9] documents containing detailed metadata describing the services offered by a particular provider.

Each service has several required operations, but it is not required that the same software or internet host computer implement all of them. For example, a WMS may offer GetMap directly but allow its GetCapabilities response to be handled by another machine on its behalf. The service metadata in Capabilities XML includes distinct online resource addresses (e.g., HTTP URLs) for each service offered.

There is no requirement in this specification dictating the format or content of those URLs; rather, this document describes only the packaging of the query parameters that must be appended to the service URL.

4.1.2 Service Taxonomy

The services provided in IP2000 (WMT2 and GFS) are shown in Table 4.1 in the corresoponding categories of the Service Taxonomy provided in ISO 19119.

Geospatial Service Taxonomy	**IP2000 Services**
Human Interaction Services	WMS Viewer Client Generator
Model/Information Management Services	Web Map Service Web Coverage Service Web Feature Service Web Registry Service Gazetteer service
Workflow/Task Services.	(none in IP2000)
Processing Services:	
- Processing services – spatial	(none in IP2000)
- Processing services – thematic	(none in IP2000)
- Processing services – temporal	(none in IP2000)
- Processing services – associations	Geoparsing service Geocoding service Geolinking service
- Processing services – metadata	(none in IP2000)
Communication Services	GML Encoding service
System Management Services	(none in IP2000)

Table 4.1 - Applying ISO 19119 Taxonomy to IP2000

This table can be considered an instance of a Services Organizer Folder (SOF) as defined in ISO 19119, although an SOF has not been defined in WMT2as an existing data type. An SOF provides a bag of service identifiers, where the identified services are grouped to support a given task or project.

4.2 Web Map Server

4.2.1 WMS Description

A Web Map Server (WMS) generates "pictures" of georeferenced data. Independent of whether the underlying data are simple features (such as points, lines and polygons) or coverages (such as gridded fields), the WMS produces an image of the data that can be directly viewed in a graphical web browser or other picture-viewing software. A WMS labels its data as one or more "Layers," each of which is available in one or more "Styles." Upon request a WMS makes an image of the requested Layer(s), in either the specified or default rendering Style(s). The image request, called GetMap, indicates the Spatial Reference System (SRS) and Bounding Box of the portion of the Earth to be mapped, and the output width, height and format of the picture. Typical output formats include Portable Network Graphics (PNG), Graphics Interchange Format (GIF), Joint Photographic Expert Group format (JPEG), Tagged Image File Format (TIFF), etc. When the data do not cover the entire field of view (such as a network of roads that includes the space between the roads), the background can be made transparent in some output formats.

A Viewer Client (VC) can issue GetMap requests for different maps to several independent Map Servers. If each map has the same geographic area and physical dimensions, and if their backgrounds are transparent, then they can be overlaid in a single window to produce a combined map. For example, server A might produce a topography image, server B a map of rivers and lakes, and server C a map of watershed boundaries. Each server maintains the type of data in which it specializes, but the end user can obtain a combined presentation of the three Layers.

4.2.2 WMS Operations

A WMS may offer the following operations:

- GetCapabilities (required)
- GetMap (required)
- GetFeatureInfo (optional)

4.2.3 SLD-Enabled WMS

WMT2 considered an extension of a traditional map server called a Styled Layer Descriptor (SLD) Map Server. Instead of generating maps of particular named layers in one ore more predefined styles, an SLD Map Server extracts features from a WFS and renders them using a stylistic description encoded in XML.

See the Styled Layer Descriptor specification[19] for more details.

An SLD WMS offers the following operations:

- GetCapabilities (required)
- GetMap (required)
- DescribeLayer (required)

4.2.4 Graphic Elements

WMT1 considered a case intermediate between the WMS and the WFS: the "Graphic Element Case." It assumes that a server containing feature data can package the data as a set of individual elements, typically in a projected reference system and with defined symbolization for geographic features, which could be directly rendered on a display device. Possible output formats include Postscript (PS), Scalable Vector Graphics (SVG), or Web Computer Graphics Metafile (WebCGM). Thus, while a map image from a WMS has a fixed size, a Graphic Element is resolution-independent and can be resized on the fly. Output formats such as PS, SVG and WebCGM are included as options in the WMS1 and WMS2 specifications, but little actual use has been made of them.

4.3 Web Feature Server

4.3.1 WFS Description

A Web Feature Server (WFS)[20] offers access to the geographic features (points, lines, polygons) in a data store. A "basic" or "read-only" WFS implements operations to describe and retrieve features, while a "transaction" WFS also implements operations to lock and modify (create, update, delete) features. A WFS Client can query the server to obtain desired features encoded as GML, display them for the user, and possibly modify them. A query is in the form of a Filter which constrains which features are returned. The ramifications of accessing a rich database of Features are considerable, and several interrelated specifications have emerged from this work: *Geographic Markup Language, Web Feature Specification, Filter Encoding Specification, Feature Identifier Specification,* and *Transaction Encoding Specification.*

4.3.2 WFS Operations

A WFS may offer the following operations:

- GetCapabilities (required)
- DescribeFeatureType (required)
- GetFeature (required)
- LockFeature (required for Transaction WFS)
- Transaction (required for Transaction WFS)

4.4 Web Coverage Server

4.4.1 WCS Description

A Web Coverage Server (WCS)[21] offers access to the actual numeric values of gridded georeferenced data or imagery. (Other, more sophisticated types of coverages are not addressed in the initial WCS spec.) A WCS client can issue a GetCoverage request to obtain these numeric values for further processing or rendering on behalf of the user. A WCS offers one or more Layers, just like a WMS, but does not render them for the user and therefore does not offer Styles. GetCoverage includes the desired SRS, Bounding Box, and output format, but does not include a Width or Height because these apply only to images. Possible output formats include GeoTIFF, NASA's Hierarchical Data Format implementation for the Earth Observing System (HDF-EOS), and an extension of GML for Coverages that has been proposed to allow

XML encoding of metadata about the coverage object and numeric values either stored inline or referenced by link to an external file.

A simple WCS Client can issue GetCoverage requests and pass the resulting geodata object to a helper application for processing and display. A more sophisticated client, or a helper application that has been retrofitted with the ability to invoke GetCoverage, can handle requests and rendering itself.

4.4.2 WCS Operations

A WCS may offer the following operations:

- GetCapabilities (required)
- GetCoverage (required)

4.5 Web Registry Server

4.5.1 WRS Description

A Web Registry Server (WRS)[22] is a catalog of OGC Web Services. It is a "stateless" catalog in that it relies on the single request/single response mechanisms of the HTTP distributed computing platform. A WRS supports registration, metadata harvesting and descriptor ingest, push and pull update of descriptors, and discovery of OGC Web Service types and instances.

4.5.2 WRS Operations

A WRS may offer the following operations:

- GetCapabilities (required)
- GetDescriptor (required)
- RegisterService (required)

4.6 GeoCoder

4.6.1 GeoCoder Description

4.6.2 GeoCoder Operations

- GetCapabilities
- GetFeature
- GeocodeFeature
- DescribeFeatureType

5 GENERAL RULES (NORMATIVE)

5.1 Version Numbering and Negotiation

5.1.1 Version Number Form

The specification version number contains up to three positive integers, separated by decimal points. The following forms are possible: "x", "x.y", or "x.y.z". Each specification (WMS, WFS, etc.) is numbered independently.

5.1.2 Version Changes

A particular specification's version number **shall** be changed with each revision. The number **shall** increase monotonically and **shall** comprise no more than three integers separated by decimal points, with the first integer being the most significant. There may be gaps in the numerical sequence. Some numbers may denote experimental or interim versions. Service instances and their clients need not support all defined versions, but **must** obey the negotiation rules below.

5.1.3 Appearance in Requests and in Service Metadata

The version number appears in at least two places: in the Capabilities XML describing a service, and in the parameter list of client requests to that service. The version number used in a client's request of a particular service instance **must** be equal to a version number which that instance has declared it supports (except during negotiation as described below). A service instance may support several versions, whose values clients may discover according to the negotiation rules.

5.1.4 Negotiation Rules

In response to a GetCapabilities request containing a version number, an OGC Web Service **must** either respond with output that conforms to that version of the specification, **or** negotiate a mutually agreeable version if the requested version is not implemented on the server. If no version number is specified in the request, the server **must** respond with the highest version it understands and label the response accordingly (for those output formats which include a protocol version number field, such as Capabilities XML).

Version number negotiation occurs as follows:

If the server implements the requested version number, the server **must** send that version.

If a version unknown to the server is requested, the server **must** send the highest version less than the requested version and label it accordingly.

2b) If the client request is for a version lower than any of those known to the server, then the server **must** send the lowest version it knows and label it accordingly.

If the client does not understand the new version number sent by the server, it **may** either cease communicating with the server **or** send a new request with a new version number that the client does understand but which is less than that sent by the server (if the server had responded with a lower version).

3b) If the server had responded with a higher version (because the request was for a version lower than any

known to the server), and the client does not understand the proposed higher version, then the client **may** send a new request with a version number higher than that sent by the server.

The process is repeated until a mutually understood version is reached, or until the client determines that it will not or cannot communicate with that particular server.

Example A: server understands versions 1, 2, 4, 5 and 8. Client understands versions 1, 3, 4, 6, and 7. Client requests version 7. Server responds with version 5. Client requests version 4. Server responds with version 4, which the client understands, and the negotiation ends successfully.

Example B: server understands versions 4, 5 and 8. Client understands version 3. Client requests version 3. Server responds with version 4. Client does not understand that version or any higher version, so negotiation fails and client ceases communication with that server.

5.2 General HTTP Request Rules

At present, the only distributed computing platform (DCP) explicitly supported by OGC Web Services is the World Wide Web itself, or more specifically Internet hosts implementing the Hypertext Transfer Protocol (HTTP)[8]. Thus the Online Resource of a service instance is an HTTP Uniform Resource Locator (URL). This URL **must** conform to the description in RFC 2616 (Section 3.2.2 "HTTP URL") [8] but is otherwise implementation-dependent; only the parameters comprising the service request itself are mandated by the OGC Web Services specifications.

HTTP supports two request operations: GET and POST. One or both of these operations may be defined for a service type and offered by a service instance, and the use of the Online Resource URL differs in each case.

5.2.1 HTTP GET

When issuing a service request using HTTP GET, a client uses the Online Resource URL as a prefix to which request parameters are appended according to the HTTP Common Gateway Interface (CGI) standard[17]: the URL is followed by a question mark character ("?") and by one or more valid name/value pairs in the form "name=value" separated by the ampersand character ("&"). As with all CGI applications, the query URL is encoded[12] to protect special characters.

Table 5.1 summarizes the components of a service request URL.

URL Component	Required/ Optional/ Experimental	Description
http://server_address/path/script	R	URL prefix of service online resource.
?	R	Separator between prefix and query.
VERSION=version	R	Request version (required except as described under Version Negotiation, above).
REQUEST=service_type	R	Request name.
UPDATESEQUENCE=number	O	The update sequence number of the service metadata used to formulate this request (allows cached metadata to be used).
PARAMETER=value	R, O, E	Standard request parameters (number and type dependent on particular request).
VSP =value	O	Vendor-specific parameters.

&	R	Separator between name=value pairs.

Table 5.1 - A general OGC Web Service Request

5.2.2 HTTP POST

When issuing a service request using HTTP POST, a client uses the Online Resource as a complete URL to which request parameters, encoded as XML, are transmitted. The details of the XML encoding differ for each request type, but in all cases parameter names are XML elements while values are XML character data (CDATA) or attributes.

This document describes general HTTP GET syntax; the POST syntax (if available) may be found in the applicable service specification document.

5.3 General HTTP Response Rules

Upon receiving a valid request, the service **must** send a response corresponding exactly to the request as detailed in the appropriate specification. Only in the case of Version Negotiation (described above) may the server offer a differing result.

Upon receiving an invalid request, the service **must** issue a Service Exception as described below.

*As a practical matter, in the WWW environment a client **should** be prepared to receive either a valid result, or nothing, or any other result. This is because the client may itself have formed a non-conforming request that inadvertently triggered a reply by something other than an OGC Web Service, because the Service itself may be non-conforming, etc.*

Response objects **must** be accompanied by the appropriate MIME type[11] for that object. For Capabilities XML, the type is "text/xml." For Service Exception, the type depends on the requested output format (see Section 5.6). All others are defined elsewhere.

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 efficiently allocate space for results, and Content-Encoding or Content-Transfer-Encoding may be necessary for proper interpretation of the results.

5.4 Service Request Parameters

5.4.1 Parameter Ordering and Case

Parameter names **shall not** be case sensitive, but parameter values **shall** be case sensitive.

Parameters in a request **may** be specified in any order.

An OGC Web Service **must** be prepared to encounter parameters that are not part of this specification. In terms of producing results per this specification, an OGC Web Service **shall** ignore such parameters.

5.4.2 Parameter Lists

Parameters consisting of lists (for example, the LAYERS and STYLES in WMS GetMap) **must** use the comma (",") as the separator between items in the list. Additional white space **must not** be used to delimit list items.

List items **may** be empty.

5.4.3 VERSION

VERSION=version specifies the protocol version number. The format of the version number, and version negotiation, are described in Section 5.1.

5.4.4 REQUEST=service_type

REQUEST indicates which service operation is being invoked, and **must** either have the value "GetCapabilities" or an operation name defined by OGC Web Services.

5.4.5 UPDATESEQUENCE=number

UPDATESEQUENCE is an optional parameter that is retrieved from the capabilities document by the client. It is envisaged that this number will be a timestamp. If the current capabilities document has a larger updateSequence number the Service **may** return an exception (with code = "ExpiredUpdateSequence").

5.4.6 FORMAT

This parameter specifies the output format of the response to an operation.

An OGC Web Service **may** offer only a subset of the formats known for that type of service, but the server **shall** advertise those formats it does support and **must** accept requests for any format it advertises. A Service Instance **may** optionally offer a new format not previously offered by other instances, with the recognition that clients are not required to accept or process an unknown format. If a request contains a Format not offered by a particular server, the server **must** throw a Service Exception (with code "InvalidFormat")

A Client **may** accept only a subset of the formats known for that type of service. If a Client and Service do not support any mutually agreeable formats, the Client may, at its discretion, cease communicating with that service, or search for an intermediary service provider that performs format conversion, or allow the user to choose other disposition methods (e.g., saving to local storage or passing to helper application).

Experimental: Possible allowed values are enumerated in the Operations section of the Capabilities XML describing a service.

> ***Experimental*** *Change from WMS 1.0: Previously, known formats were enumerated in a %KnownFormats entity in the Capabilities XML DTD.*

5.4.7 EXCEPTIONS=exception_format

The format in which to report errors. See Section 5.6 on Service Exceptions, below.

5.4.8 SRS

The Spatial Reference System (SRS) is a text parameter that names a horizontal coordinate reference system code. The name includes a namespace prefix, a colon, a numeric identifier, and possibly a comma followed by additional parameters. This specification defines two namespaces, EPSG and AUTO, which are defined below.

OGC Web Services are **not** required to support all possible SRSes, but **shall** advertise in their Capabilities XML those projections which they do offer and **must** accept requests for all advertised projections. If a request contains an SRS not offered by a particular server, the server **must** throw a Service Exception (code = "InvalidSRS").

Clients are **not** required to support all possible SRSes. If a Client and Service do not support any mutually agreeable SRS, the Client may, at its discretion, cease communicating with that service, or search for an intermediary service provider that performs coordinate transformations, or allow the user to choose other disposition methods.

5.4.8.1 EPSG Namespace for SRS

The EPSG namespace makes use of the European Petroleum Survey Group tables [13], which define numeric identifiers (the EPSG "horiz_cs" code) for many common projections and which associate projection or coordinate metadata (such as measurement units or central meridian) for each identifier. For example, EPSG code 4326 refers to WGS84 latitude/longitude coordinates in degrees with Greenwich as the central meridian. An SRS name in the EPSG namespace includes only the prefix and the identifier (e.g., "EPSG:4326"). This format is used both as the value of the SRS parameter in a service request and as the value of an <SRS> element in the Capabilities XML.

5.4.8.2 AUTO Namespace for SRS

The AUTO namespace is used for "automatic" projections; that is, a class of projections that includes an arbitrary central meridian [14]. An SRS name in the AUTO namespace includes an identifier, a code indicating what units are to be used for bounding boxes in that SRS, and values for the central longitude and latitude: "AUTO:wm_id,epsg_units,lon,lat". wm_id is one of the projection IDs defined by this specification in [14]; epsg_units is one of the EPSG-defined [13] numbers for units, and lon,lat are longitude,latitude in decimal degrees. For example,
"AUTO:42003,9001,-100,45" is auto orthographic projection, bounding box in meters, center at 100 West, 45 North. In addition to the Auto Orthographic projection, Auto UTM (Universal Transverse Mercator), Auto Transverse Mercator (with the given central meridian), and Auto Equirectangular (which uses the given central latitude to scale the X axis) are defined. See reference [14] for additional details.

In a request for a georeferenced map or data, the complete AUTO SRS is specified including latitude, longitude, and units. In Capabilities XML, the latitude, longitude, and units variables are omitted.

5.4.8.3 Undefined SRS

A Server may offer geographic information whose precise spatial reference is undefined. For example, a digitized collection of hand-drawn historical maps may represent an area of the Earth but not employ a modern coordinate system. In such case, the value "NONE" **must** be used when declaring the SRS of such a collection or object. Clients **should not** attempt to overlay information whose SRS=none with other information.

5.4.9 BBOX

The Bounding Box (BBOX) is a set of four comma-separated decimal, scientific notation, or integer values (if integers are provided where floating point is needed, the decimal point is assumed at the end of the number). These values specify the minimum X, minimum Y, maximum X, and maximum Y ranges, in that order, expressed in units of the SRS of the request such that a rectangular area in the units of the SRS is defined. Where the SRS units are not units of linear measure but are expressed in latitude and longitude, X is taken to mean Longitude and Y is taken to mean Latitude. The four bounding box values indicate the outside edges of a rectangle, as in Figure 5.1: minimum X is the left edge, maximum X the right, minimum

Y the bottom, and maximum Y the top. The relation of the BBOX to the image pixel matrix is shown in the figure: the bounding box goes around the "outside" of the pixels of the image rather than through the centers of the border pixels. In this context, individual pixels have an area.

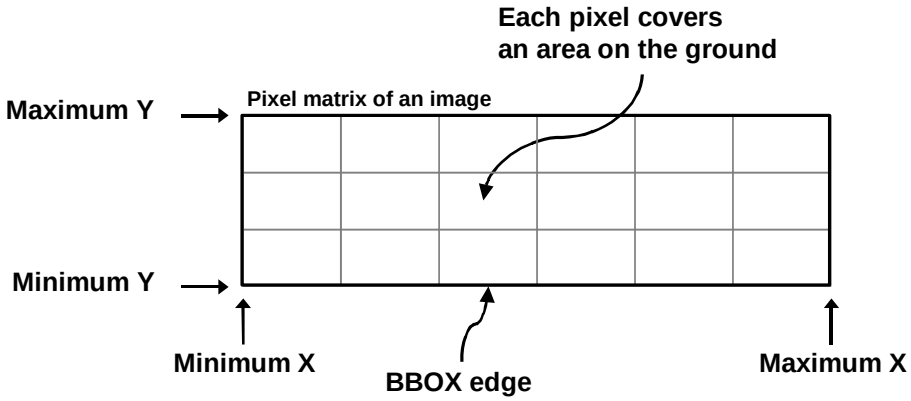

Figure 5.1 - BBOX representation

A BBOX **should not** have zero area.

If a request contains an invalid BBOX (e.g.., one whose minimum X is greater than or equal to the maximum X, or whose minimum Y is greater than or equal to the maximum Y) the server **may** throw an exception.

If a request contains a BBOX whose area does not overlap at all with the BoundingBox advertised in the Capabilities XML for the requested geodata object, the server **should** return empty content (e.g., a blank map, empty coverage file, null feature set) for that element. Any elements that are partly or entirely contained in the BBOX **should** be returned in the appropriate format.

If the BBOX values are not defined for the given SRS (e.g., latitudes greater than 90 degrees in EPSG:4326), the server **should** issue a Service Exception (code = "InvalidBoundingCoordinates".

5.4.10 Additional PARAMETER=value pairs

All service requests except GetCapabilities require additional parameters to unambiguously state what result to construct. For example, a GetMap request includes the desired size of the map image. Each OGC Web Service specification indicates the relevant parameters for its operation(s).

5.4.11 Vendor-Specific Parameters

Finally, the requests allow for optional vendor-specific parameters (VSPs) which will enhance the results of a request. Typically, these are used for private testing of non-standard functionality prior to possible standardization. A generic client is **not** required or expected to make use of these VSPs.

An OGC Web Service **must** produce a valid result even if VSPs are missing or malformed (i.e., the Service **shall** supply a default value), or if VSPs are supplied that are not known to the Service (i.e., the Service **shall** ignore unknown parameters in the URL).

An OGC Web Service **may** choose not to advertise some or all of its VSPs. Clients **may** read the internal DTD and formulate requests using any VSPs advertised therein.

Vendors **should** choose parameter names with care when defining vendor-specific parameters to avoid name clashes with standard parameters.

Experimental: An OGC Web Service **may** declare VSPs within the Operations section of its Capabilities XML. All VSPs **must** be declared optional in the Operations metadata.

> ***Experimental*** *Change from WMS 1.0: Previously, Vendor Specific Parameters were defined within an internal DTD section of the Capabilities XML response.*

5.5 Service Result

The return value of a Service request **shall** be the same as the type requested in the FORMAT parameter. In an HTTP environment, the MIME type of the returned value **shall** correspond to the return value **unless** there is no MIME type for that value. If a Service Exception occurs, the return type shall instead be that specified in the EXCEPTIONS parameter.

5.6 Service Exceptions

Upon receiving a request which is invalid according to the rules of the Distributed Computing Platform (DCP) in use, the service **may** issue an exception of a type valid in that DCP (e.g., in the HTTP DCP, if the URL itself is wrong an HTTP 404 error code is sent).

Upon receiving a request which is invalid according to the relevant OGC Web Services specification, the service **must** issue a Service Exception as defined here. The Exception is meant to describe to the client application or its human user the reason(s) that the request is invalid

The EXCEPTIONS parameter in a request dictates how a Service instance shall behave if it encounters an error. The error can be due to a malformed request or to other circumstances. However, Services should attempt to describe the problem using the Exception format given in the request.

The only value of the EXCEPTIONS parameter that is defined for all OGC Web Services is "SE_XML," which means "Service Exception XML." Particular services may define other formats; for example, a WMS may offer to write the text of the error message in an image via the "INIMAGE" format.

Note that a client should also be prepared for other returned values and types since there is a possibility that the Service instance is poorly behaved or that a request was directed at a non-compliant OGC Web Service.

5.6.1 SE_XML format

If an exception occurs, the server is asked to respond with an XML document described by the Service Exception DTD in Appendix A.3. In an HTTP environment, the MIME type of the returned XML **must** be "text/xml". Specific error messages can be included either as chunks of plain text or as XML-like text containing angle brackets ("<" and ">") if included in a character data (CDATA) section as shown in the example of Service Exception XML in Appendix A.3.1.

Service Exceptions **may** include error codes as specified. Services **shall not** use these codes for purposes other than those specified. Specific service interface specifications will define the content and meaning of additional Exception Codes. Clients **may** use these codes to automate responses to Service Exceptions.

The specific codes and semantics of allowed exceptions may be extended by each specification.

6 THE GETCAPABILITIES OPERATION (NORMATIVE)

A critical requirement for interoperability is that each service describe itself in a standardized fashion. All OGC Web Services must offer service metadata packaged in an Extensible Markup Language (XML) document. The GetCapabilities operation is used to request this metadata.

6.1 General

The GetCapabilities operation is designed to provide clients of OGC Web Services with a machine-readable description of particular service instances. The description includes general information about the service, what operations that service supports, and what content it offers.

An OGC Web Service instance **must either** implement the GetCapabilities operation **or** ensure that there exists another web server that provides the Capabilities response on the Service's behalf. If there is no means of a client accessing the capabilities XML, then the service instance is not a conforming OGC Web Service.

Internally, the Service **may** choose either to generate the GetCapabilities response dynamically **or** to return a static XML file in response to the GetCapabilities request.

6.2 GetCapabilities Request

The operation has two input parameters – the request name and the request version. Table 6.1 shows the parameters of a GetCapabilities request.

Request Parameter	Required/ Optional/ Experimental	Description
REQUEST=GetCapabilities	R	Request name
SERVICE=service	R	Service type
VERSION=version	O	Request version

Table 6.1 - The parameters of a GetCapabilities Request.

6.2.1 GetCapabilities Request Parameters

6.2.1.1 VERSION

This parameter is discussed in the Service Request Parameters section, above. It is optional to allow version negotiation as described earlier, but **should** be specified by clients wishing to receive a particular version.

6.2.1.2 REQUEST

This nature of this parameter is specified in the Basic Services Model document. For the GetCapabilities operation, the value "GetCapabilities" **must** be used.

Change from WMS 1.0: The value of this parameter was previously "capabilities".

6.2.1.3 SERVICE

This experimental parameter indicates the type of Service for which Capabilities information is requested. It allows the same interface (e.g., the same URL prefix) to offer several different services. The allowed values are as follows:

- WMS (describe WMS at this interface)
- WFS (describe WFS at this interface)
- WCS (describe WCS at this interface)
- LIST (list only the available services at this interface)

Change from WMS 1.0: SERVICE is a new parameter.

6.3 GetCapabilities Response - DTD (Normative)

The Service Metadata output **shall** be in the form of XML, which **must** be valid against the XML Document Type Definition (DTD) defined by the appropriate service's interface specification. At present, each service type offers its own version of Capabilities XML. An eventual goal is to bring these disparate XML responses into a common framework as defined in the following Experimental section.

The service metadata schema provided below in XML is an implementation of the abstract model for service metadata provided in ISO 19119 Geographic Information – Services.

6.4 GetCapabilities Response - Schema (Experimental)

The Service Metadata output **shall** be in the form of XML, which **must** be valid against the XML Schema defined for the SERVICE named in the request. The Service List Schema is defined in Appendix A.2, below. The Schema for a particular service includes a Service-Independent part discussed below and defined in Appendix A.1, followed by a Service-Specific part defined in the relevant specification document.

Change from WMS 1.0: Previously, an XML Document Type Definition (DTD) was used instead of XML Schema to specify the structure of Capabilities XML.

There may exist several numbered versions of each service's XML Schemas; the version to return depends on the value of the VERSION parameter and on the version negotiation rules discussed above.

If the GetCapabilities request does not return XML which can be validated against the appropriate Schema, then the Service instance **is not** a conforming OGC Web Service. If there is no means of a client accessing the Capabilities XML, then the Service instance **is not** a conforming OGC Web Service.

The MIME type [11] of the returned XML **shall** be "text/xml". (Note that text/xml is favored over application/xml by RFC 2376 [16] when the content of the message is mainly plain text [e.g., no 16-bit chars, but with HTTP even 16-bit Unicode (UTF-16) is permitted].)

6.3.1 Basic Service Metadata and Service-Specific Metadata

The *structure* of OGC Web Services metadata includes three components: a section of basic service metadata that is relevant for all services, and two sections of additional metadata specific to that type of service. Those two sections describe the operations and content available from a particular service. The basic service metadata has the structure defined by the XML Schema in Appendix A.1. An actual service will define its own Schema, which will begin with the contents of the basic service Schema and add elements as necessary. The basic service metadata can be thought of as an abstract class which is inherited and extended by actual services. The abstract elements for which each service is expected to define valid values include the following common elements.

6.3.1.1 Service

The <service> section of the Capabilities XML provides general information about an OGC Web Service. The content includes (but is not limited to) such elements as:

- type of service offered (using a vocabulary of types defined by OGC)
- human-readable title of the service
- abstract and keywords providing further description
- point of contact information
- access restrictions involving payment, security, or usage, if any.

6.3.1.2 Operations

The <operations> section of the Capabilities XML provides detailed information about the particular operations available from this service instance. One or more operations are listed; for each, information including (but not limited to) the following is provided:

- name of the operation
- human-readable description
- list of one or more request parameters
- for each parameter, possible lists of valid values or an indication of where such values may be found in the Content section.

The Operations section may contain pointers into the Content section, as described below.

6.3.1.3 Content

The <service> section of the Capabilities XML holds information describing the geodata exposed by the OGC Web Service instance. For example, a WFS exposes a collection of features, while a WMS exposes map layers.

Where "feature types" are referenced the feature type reference will be the URI of a network addressable feature schema. If the identifier is an unqualified identifier then its schema will be available via a DescribeFeatureType request.

6.3.2 Pointers from Operations to Content

The Operations metadata may include references to the Content section that indicate where valid parameter values may be found. The syntax for such pointers is taken from the XPath specification [18] from the World Wide Web Consortium. All pointers start with "/content" to indicate that the <content> section is

being referenced. Following that are one or more names of elements found in the <content> XML in order of their appearance in the XML tree. For example, the pointer "/content/subelement/subsub" refers to the content any <subsub> element which is a child of a <subelement> which is a child of <content>:

```
<content>
  <subelement>
    <subsub>Target of reference</subsub>
  </subelement>
</content>
```

[what about attribute values, or target elements containing other elements?]

6.3.3 Quality of service

QUALITY is an optional Operation parameter that a consists of a list of available "quality of service" options for a service instance. The available quality of service options are defined in a <QOSOptions> element within the <content> metadata. Each QOS option contains the subelements Name, Title and Abstract, all of which are defined by the service instance. Name defines the parameter values which can be encoded in the operation request. Title provides a human-readable short title for such a list. Abstract will contain human-readable text which can be used to provide an explanation of the nature of the quality differences. Additional elements can be defined by particular services.

For example, consider an address geocoder. In the Operations metadata, the following might be included. This portion would be identical for all service instances.

```
<parameter repeatable="true" optional="yes" direction="in">
  <parameterName>
    <nameValue>QUALITY</nameValue>
    <nameNameSpace>OGC</nameNameSpace>
  </parameterName>
  <parameterType type="string" />
  <permittedValues>
    <contentInstance type="extend">/content/QOSOptions/QOS/Name</contentInstance>
  </permittedValues>
</parameter>
```

In the Content metadata, the service instance would define specific QOS options within the following structure:

```
<QOSOptions>
<QOS>
  <Name>EXACT</Name>
  <Title>Find exact matches</Title>
  <Abstract>A match is made if street number is found in the range of numbers on a street
with an identical name within the specified ZIP code</Abstract>
</QOS>
 <QOS>
  <Name>FUZZY</Name>
  <Title>Find close matches</Title>
<Abstract>If no exact match for a street name is found, similar names within the same Zip
Code are tested to see if they contain the street number within the address
range.</Abstract>
</QOS>
</QOSOptions>
```

7 REFERENCES

[8] Fielding et. al., "RFC 2616: Hypertext Transfer Protocol – HTTP 1/1," Internet Society Network Working Group, June 1999, http://www.ietf.org/rfc/rfc2616.txt.

[9] Bray, Paoli, Sperberg-McQueen, eds., "Extensible Markup Language (XML) 1.0 (Second Edition)", World Wide Web Consortium, October 2000, http://www.w3.org/TR/2000/REC-xml-20001006.

[11] Borenstein N., and N. Freed, "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies", RFC 1521, Bellcore, Innosoft, September 1993, http://www.ietf.org/rfc/rfc1521.txt.

[12] Berners-Lee, T., Fielding, N., and Masinter, L., "Uniform Resource Identifiers (URI): Generic Syntax", IETF RFC 2396, http://www.ietf.org/rfc/rfc2396.txt.
[13] European Petroleum Survey Group, "EPSG Geodesy Parameters", http://www.petroconsultants.com/products/geodetic.html.
[14] Web Mapping Testbed, "WM Automatic Projections", http://www.digitalearth.gov/wmt/auto.html
[15] Bradner, Scott, "RFC 2119: Key words for use in RFCs to Indicate Requirement Levels," March 1997, http://www.ietf.org/rfc/rfc2119.txt
[16] Whitehead, E. et. Al., "XML Media Types," July 1998, http://www.ietf.org/rfc/rfc2376.txt
[17] National Center for Supercomputing Applications, "The Common Gateway Interface," undated, http://hoohoo.ncsa.uiuc.edu/cgi/
[18] Clarke, J. and DeRose, S. (eds.), "XML Path Language (XPath) Version 1.0," World Wide Web Consortium Recommendation, November 1999, http://www.w3.org/TR/xpath/.
[19] Cuthbert, A. (ed.), "OpenGIS Discussion Paper #01-028: Styled Layer Descriptor Draft Candidate Implementation Specification 0.7.0," February 2001, http://www.opengis.org/techno/discussions.htm.
[20] Vretanos, P. (ed.), "OpenGIS Discussion Paper #01-023: Web Feature Server Draft Candidate Implementation Specification 0.0.12," January 2001, http://www.opengis.org/techno/discussions.htm.
[21] Evans, J. (ed.), "OpenGIS Discussion Paper #01-018: Web Coverage Server Draft Candidate Implementation Specification 0.4," February 2001, http://www.opengis.org/techno/discussions.htm.
[22] Reich, L. (ed.), "OpenGIS Document #01-024: Web Registry Server Draft Candidate Implementation Specification 0.0.2", January 2001, unpublished.
[23] Margoulies, S. (ed.), "OpenGIS Document #01-026: GeoCoder Service Draft Candidate Implementation Specification 0.7.5," February 2001, unpublished.
[24] ISO 8601:1988(E), "Date/Time Representations," http://www.iso.ch/markete/8601.pdf.
[25] ISO 191119 CD2, "Geographic Information - Services", ISO TC211 Document 1044, Date: 2001-01-29.

APPENDIX A: XML DTDS AND SCHEMA

This annex contains XML Schema and sample XML for Service-Independent Metadata, the Service List Response, and Service Exceptions. The Service Exceptions section is Normative; others are Experimental.

A.1 Basic Service Metadata Schema (Experimental)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 (http://www.xmlspy.com) by John Evans (private)-->
<!--W3C Schema generated by XML Spy v3.5 (http://www.xmlspy.com)-->
<xsd:schema xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema" elementFormDefault="qualified">
  <xsd:import namespace="http://www.w3.org/1999/xlink"
schemaLocation="http://www.opengis.net/namespaces/gml/core/xlinks.xsd"/>
  <xsd:element name="OGC_Web_Service">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="service">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="serviceType">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element ref="nameValue"/>
                    <xsd:element ref="nameNameSpace"/>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
              <xsd:element name="title" type="xsd:string"/>
              <xsd:element name="abstract" type="xsd:string" minOccurs="0"/>
              <xsd:element name="keywords" minOccurs="0">
```

```
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="keyword" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
                <xsd:element name="typeCode" minOccurs="0">
                  <xsd:complexType>
                    <xsd:attribute name="KeyType" use="required">
                      <xsd:simpleType>
                        <xsd:restriction base="xsd:NMTOKEN">
                          <xsd:enumeration value="discipline"/>
                          <xsd:enumeration value="place"/>
                          <xsd:enumeration value="stratum"/>
                          <xsd:enumeration value="temporal"/>
                          <xsd:enumeration value="theme"/>
                        </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:attribute>
                  </xsd:complexType>
                </xsd:element>
                <xsd:element name="thesaurusName" type="xsd:string" minOccurs="0"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="linkage" xlink:simpleLink=""/>
          <xsd:element name="pointOfContact">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="individualName" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
                <xsd:element name="organisationName" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
                <xsd:element name="positionName" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
                <xsd:element name="contactInfo" minOccurs="0" maxOccurs="unbounded">
                  <xsd:complexType>
                    <xsd:sequence>
                      <xsd:element name="phone" minOccurs="0">
                        <xsd:complexType>
                          <xsd:sequence>
                            <xsd:element name="voice" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
                            <xsd:element name="facsimile" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
                            <xsd:element name="other" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
                            <xsd:element name="otherType" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
                          </xsd:sequence>
                        </xsd:complexType>
                      </xsd:element>
                      <xsd:element name="address" minOccurs="0">
                        <xsd:complexType>
                          <xsd:sequence>
                            <xsd:element name="deliveryPoint" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
                            <xsd:element name="city" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="administrativeArea" type="xsd:string"
minOccurs="0"/>
                            <xsd:element name="postalCode" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="country" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="electronicMailAddress" type="xsd:string"
minOccurs="0" maxOccurs="unbounded"/>
                          </xsd:sequence>
                        </xsd:complexType>
                      </xsd:element>
                      <xsd:element name="onLineResource" type="xsd:string" minOccurs="0"/>
                      <xsd:element name="hoursOfService" type="xsd:string" minOccurs="0"/>
                      <xsd:element name="contactInstructions" type="xsd:string"
minOccurs="0"/>
                    </xsd:sequence>
                  </xsd:complexType>
                </xsd:element>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
```

```
          <xsd:element name="accessProperties" minOccurs="0">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="fees" type="xsd:string" minOccurs="0"/>
                <xsd:element name="plannedAvailableDateTime" type="xsd:string"
minOccurs="0"/>
                <xsd:element name="orderingInstructions" type="xsd:string" minOccurs="0"/>
                <xsd:element name="turnaround" type="xsd:string" minOccurs="0"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="legalConstraints" minOccurs="0">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element ref="useLimitation" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="propertyRightsCode" minOccurs="0" maxOccurs="unbounded">
                  <xsd:complexType>
                    <xsd:attribute name="Restrict" use="required">
                      <xsd:simpleType>
                        <xsd:restriction base="xsd:NMTOKEN">
                          <xsd:enumeration value="copyright"/>
                          <xsd:enumeration value="patent"/>
                          <xsd:enumeration value="patentPending"/>
                          <xsd:enumeration value="license"/>
                          <xsd:enumeration value="intellectualPropertyRights"/>
                          <xsd:enumeration value="trademark"/>
                        </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:attribute>
                  </xsd:complexType>
                </xsd:element>
                <xsd:element name="useConstraintsCode" minOccurs="0" maxOccurs="unbounded">
                  <xsd:complexType>
                    <xsd:attribute name="Restrict" use="required">
                      <xsd:simpleType>
                        <xsd:restriction base="xsd:NMTOKEN">
                          <xsd:enumeration value="copyright"/>
                          <xsd:enumeration value="patent"/>
                          <xsd:enumeration value="patentPending"/>
                          <xsd:enumeration value="license"/>
                          <xsd:enumeration value="intellectualPropertyRights"/>
                          <xsd:enumeration value="trademark"/>
                        </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:attribute>
                  </xsd:complexType>
                </xsd:element>
                <xsd:element name="otherConstraints" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="securityConstraints" minOccurs="0">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element ref="useLimitation" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="classificationCode" minOccurs="0">
                  <xsd:complexType>
                    <xsd:attribute name="Classification" use="required">
                      <xsd:simpleType>
                        <xsd:restriction base="xsd:NMTOKEN">
                          <xsd:enumeration value="unclassified"/>
                          <xsd:enumeration value="codeWord"/>
                          <xsd:enumeration value="confidential"/>
                          <xsd:enumeration value="secret"/>
                          <xsd:enumeration value="restricted"/>
                          <xsd:enumeration value="topsecret"/>
                        </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:attribute>
                  </xsd:complexType>
                </xsd:element>
                <xsd:element name="userNote" type="xsd:string" minOccurs="0"/>
                <xsd:element name="classificationSystem" type="xsd:string" minOccurs="0"/>
                <xsd:element name="handlingDescription" type="xsd:string" minOccurs="0"/>
```

```
            </xsd:sequence>
           </xsd:complexType>
          </xsd:element>
         </xsd:sequence>
        </xsd:complexType>
       </xsd:element>
       <xsd:element name="operations">
        <xsd:complexType>
         <xsd:sequence>
          <xsd:element name="operationMetadata" minOccurs="0" maxOccurs="unbounded">
           <xsd:complexType>
            <xsd:sequence>
             <xsd:element name="operationName">
              <xsd:complexType>
               <xsd:sequence>
                <xsd:element ref="nameValue"/>
                <xsd:element ref="nameNameSpace"/>
               </xsd:sequence>
              </xsd:complexType>
             </xsd:element>
            </xsd:sequence>
           </xsd:complexType>
          </xsd:element>
         </xsd:sequence>
        </xsd:complexType>
       </xsd:element>
       <xsd:element name="content">
        <xsd:complexType/>
       </xsd:element>
      </xsd:sequence>
      <xsd:attribute name="version" type="xsd:string" use="fixed" value="0.0.4"/>
      <xsd:attribute name="updateSequence" type="xsd:string" use="default" value="0"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="nameNameSpace" type="xsd:string"/>
  <xsd:element name="nameValue" type="xsd:string"/>
  <xsd:element name="useLimitation" type="xsd:string"/>
</xsd:schema>
```

A.1.1 Basic Service Metadata XML Example (Experimental)

```
<?xml version='1.0' encoding="UTF-8" standalone="no" ?>
<!DOCTYPE OGC_Web_Service SYSTEM
 "http://www.digitalearth.gov/wmt/xml/service_0_0_4.dtd" >

<OGC_Web_Service version="0.0.4" updateSequence="0">
  <service>
    <serviceType>
      <nameValue>OGC_Web_Service</nameValue>
      <nameNameSpace>OGC</nameNameSpace>
    </serviceType>
    <title>Generic OGC Web Service</title>
    <abstract>This XML will be subclassed and extended by
              specific OGC Web services.</abstract>
    <keywords>
      <keyword>word1</keyword>
      <keyword>word2</keyword>
      <keyword>multi-word phrase</keyword>
    </keywords>
    <!-- Top-level web address of service or service provider.
         URLs for particular operations are listed in <Operations>, below.-->
    <linkage xlink:href="Web_Site_URL"
     xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"  />
    <!-- Contact information -->
    <pointOfContact>
      <individualName>Jeff deLaBeaujardiere</individualName>
      <organisationName>UMBC/NASA GEST</organisationName>
      <positionName>Computer Scientist</positionName>
      <contactInfo>
      <phone>
        <voice>+1 301 286-1569</voice>
        <facsimile>+1 301 286-1777</facsimile>
      </phone>
        <address>
```

```
          <deliveryPoint>NASA Goddard, Code 900.3</deliveryPoint>
          <city>Greenbelt</city>
          <administrativeArea>MD</administrativeArea>
          <postalCode>20771</postalCode>
          <country>USA</country>
          <electronicMailAddress>delabeau@iniki.gsfc.nasa.gov</electronicMailAddress>
        </address>
      </contactInfo>
    </pointOfContact>
    <!-- Fees or access constraints imposed. -->
    <accessProperties>
      <fees>none</fees>
    </accessProperties>
    <legalConstraints>
      <useLimitation>none</useLimitation>
    </legalConstraints>
    <securityConstraints>
      <useLimitation>none</useLimitation>
    </securityConstraints>
  </service>

  <!-- placeholder for list of operations (e.g., GetMap or DescribeFeature) -->
  <operations>
    <!-- zero or more operationMetadata blocks -->
    <operationMetadata>
      <operationName>
        <nameValue></nameValue>
        <nameNameSpace>OGC</nameNameSpace>
      </operationName>
    </operationMetadata>
  </operations>

  <!-- placeholder for enumeration of content (e.g., Layers or FeatureTypeList) -->
  <content></content>

</OGC_Web_Service>
```

A.2 Service List Schema (Experimental)

To be written.

A.3 Service Exception DTD (Normative)

```
<!ELEMENT ServiceExceptionReport (ServiceException*)>
<!ATTLIST ServiceExceptionReport version CDATA #REQUIRED>

<!ELEMENT ServiceException (#PCDATA)>
<!ATTLIST ServiceException code CDATA #IMPLIED>
```

A.3.1 Sample Service Exception XML (Informative)

```
<?xml version='1.0' encoding="UTF-8" standalone="no" ?>
<!DOCTYPE ServiceExceptionReport SYSTEM
"http://www.digitalearth.gov/wmt/xml/exception_1_0_1.dtd">
<ServiceExceptionReport version="1.0.1">
<ServiceException code="Unspecified">
Plain text message about an error.
</ServiceException>
<ServiceException code="ExpiredUpdateSequence">
new service metadata available
</ServiceException>
<ServiceException>
<![CDATA[
Error in module <foo.c>, line 42

A message that includes angle brackets in text
must be enclosed in a Character Data Section
as in this example.  All XML-like markup is
ignored except for this sequence of three
```

```
closing characters:
]]>
</ServiceException>
<ServiceException>
<![CDATA[
<Module>foo.c</Module>
<Error>An error occurred</Error>
<Explanation>Similarly, actual XML
can be enclosed in a CDATA section.
A generic parser will ignore that XML,
but application-specific software may choose
to process it.</Explanation>
]]>
</ServiceException>
</ServiceExceptionReport>
```

APPENDIX B: FORMATTING DATES AND TIMES (NORMATIVE)

B.1 Overview

This appendix describes extensions to the ISO8601 specification for denoting moments and periods in time. These allow OGC Web Services to support temporal data descriptions and requests. This specification is based on ISO 8601:1988(E) Date/Time Representations [24] for date+time and for periods of time; it also extends ISO 8601 to allow years before 1 AD. It details the following topics:

B.2. Time Format Details

B.3. Period Format

B.4. Date Fragments

  B.4.1. Truncated representation

  B.4.2. Days of week

B.5. Examples

  B.5.1. Complete dates and times

  B.5.2. Date and time fragments

B.2 Time Format Details

Uses ISO 8601 "extended" format:

B.2.1 Syntax

Up to 14 digits, and optionally a decimal point followed by zero or more digits, specifying century, year, month, day, hour, minute, seconds, decimal seconds, with non-numeric characters to separate each piece:

ccyy-mm-ddThh:mm:ss.sssZ

(ISO 8601 prefers a decimal comma but allows a decimal period.) All times are assumed to be UT, hence the trailing Z for "zulu."

The precision may be reduced by omitting least-significant digits. Thus, 4 digits specify the year, 6 the year and month, 8 the year-month-day, etc:

ccyy

ccyy-mm

ccyy-mm-dd

ccyy-mm-ddThh

B.2.2 Extension for years B.C.E.

A non-ISO extension is proposed to handle years B.C.E (before year 0001 AD). Because a leading hyphen has meaning in ISO 8601 (it can indicate missing parts of the time), preceding times with a minus sign is not advisable. Instead, the prefix 'B' is used to indicate B.C.E. years. For example:

B1350 (1350 "BC", approximate time of Tutankhamen).

B.2.3 Extension for geologic datasets

A non-ISO extension is proposed to handle geologic datasets refering to the distant past. Geologists refer to times some number of thousand, million or billion (10^9) years ago as ka, Ma or Ga. Thus, the prefixes K, M and G, followed by an integer or floating-point number of years, are proposed. For example:

M150 (Jurassic period, 150 Ma)

K150000 (Jurassic period, 150 Ma)

K18 (last Ice Age, 18 ka)

M0.018 (last Ice Age, 18 ka)

G5 (Earth's crust solidifies, 5 Ga)

B.3 Period Format

A Period indicates the time resolution of the available data, or the frame interval in a requested animation. The ISO 8601 specification for representing a duration of time is used: Designator P (for Period), number of years Y, months M, days D, time designator T, number of hours H, minutes M, seconds S. Unneeded elements may be omitted. For example:

P1Y -- 1 year

P1M10D -- 1 month plus 10 days

PT2H -- 2 hours

PT1.5S -- 1.5 seconds

B.4 Date Fragments

In some contexts, date information may be incomplete. For example, a Geoparser service that scans text documents for references to locations or times may encounter partial references like "Tuesday" or "August 30." To allow this time specification to handle such fragments, we include the "truncated representation" allowed by ISO8601, and we add 7 terms to replace days of the week.

B.4.1 Truncated representation

The "truncated representation" allowed by ISO8601 uses hyphens to denote pieces missing from the most significant end of the time string. Add one hyphen for each missing two-digit century, year, month, day, hour or minute. See examples in 5.2, below.

B.4.2 Days of the week

Add 7 terms to replace days of the week: 'MON', 'TUE', 'WED', 'THU', 'FRI', 'SAT', 'SUN'. These optional terms appear at the start of the date/time string, and a comma separates the weekday from the ISO8601 string. If the date fragment contains only a day of week, then the comma is not necessary. See examples in 5.2, below.

B.5 Examples

B.5.1 Complete dates and times

?? A single moment (scalar value):

2000-06-23T20:07:48.11

?? Quarterly data (comma-separated list):

1999-01-01,1999-04-01,1999-07-01,1999-10-01

?? Daily data taken at noon since April 15 1995 (periodic interval):

1995-04-22T12:00/2000-06-21T12:00/P1D

?? Current data refreshed every 30 min (periodic interval):

2000-06-18T14:30/2000-06-18T14:30/PT30M

B.5.2 Date and time fragments

?? --11-21 (Nov. 21, truncated century and year, omitted hh:mm:ss)

?? TUE,--11-21 (Tuesday, Nov. 21)

?? TUE (Tuesday)

?? 2000 (year 2000)

?? -99 (year 99, century not given)

?? T14 (2pm local time)

?? MON,---21T14 (Monday the 21st at 2pm)

?? T14Z (2pm UTC)

?? T14-04:00 (2pm EDT, 4hrs earlier than UTC)

T-10 (ten minutes past an unspecified hour)

APPENDIX C: DESCRIBING MULTI-DIMENSIONAL DATA (NORMATIVE)

C.1 Overview

This appendix describes support for multi-dimensional data in the service metadata and operation request parameters of OGC Web Services. This specification includes the following categories:

1. Declaring Dimensions

2. Specifying Extents / Bounds and spatio-temporal resolution

3. Opaque and "Indestructible" Data

4. GetMap and GetCoverage Requests

C.2 Declaring Dimensions

A new element, Dimension, in the Capabilities XML response contains generic dimension names and measurement units. (These are used to define the extents, or bounds, of Map, Coverage or Feature Layers, as described below.) The following is a DTD fragment expression for the Dimension element:

[Replace with Schema fragment]

```
<!Element Dimension EMPTY>
<!ATTLIST Dimension
name         ID       #REQUIRED
units        CDATA    #REQUIRED
unitSymbol   CDATA    #IMPLIED>
```

The following are some examples of server-defined dimensions:

```
<Dimension name="wavelength" units="Angstrom" unitSymbol="Ao" />
<Dimension name="temperature" units="Kelvin" unitSymbol="K" />
<Dimension name="pressure_scale_height" units="millibar" unitSymbol="mbar" />
<Dimension name="altitude" units="kilometer" unitSymbol="km" />
```

Dimension elements must comply with the following three rules:

?? Because the Dimension name values are used in GET requests, they must be case-insensitively unique within a Layer and should contain no whitespace.

?? Unit names and abbreviations are from the Unified Code for Units of Measure (UCUM) <http://aurora.rg.iupui.edu/~schadow/units/UCUM/>. The required units attribute is from the UCUM "name" column. The optional (but recommended) unitSymbol is from the UCUM "c/s" column (case-sensitive abbreviation). Prefix names and symbols may be present as well.

?? All Dimensions in a Capabilities XML response are server-defined, with two exceptions. The dimensions named time and elevation are privileged special cases, predefined as follows:

```
<Dimension name="elevation" units="EPSG:4301" />
<Dimension name="time" units="ISO8601" />
```

If a server opts to specify elevation or time values in reference systems other than those listed above, it must use other dimension names instead of "elevation" and "time" (e.g., "altitude" and "date").

Notes:

?? *EPSG Vertical Datum Codes are listed at http://www.petroconsultants.com/products/geodetic.html*

?? *The ISO8601-based time representation is detailed in Appendix B, above.*

C.3 Specifying Extents and Spatio-Temporal Resolution

A server's Capabilities XML response **may** describe each geodata object's domain (the portion of space and time it can provide values for) and its range (the set of values that these locations can take). This information can appear in an Extent element or in a BoundingBox.

For one-dimensional reference systems, this information appears in an Extent element. For 3-D or 4-D reference systems, the 2-D BoundingBox of WMS 1.0 has new attributes to encode vertical and temporal extent and to indicate the data resolution along any of the four dimensions.

C.3.1 Bounds and resolution along a single dimension: Extent

The Capabilities XML response uses zero or more Extent elements to declare the bounds of a geodata object along along zero more or independent single dimensions. For example, a satellite image may be available at several different times and several different wavelength bands. The times and bands can each be separately declared using Extent.

Format of Extent element:

<Extent name="*dimension_name*" default="*default_value*">
extent_value
</Extent>

?? *dimension_name* is the name of a Dimension (<Dimension name="dimension_name" …>) previously defined for that geodata object (as described in the preceding section). The Dimension name and Extent name **must** match exactly (case-sensitively).

?? *default_value* declares what value would be used along that Dimension if a Web request omits a constraint in that Dimension. A 'default' is **optional** but **strongly recommended**. If a request does not include a constraint along this dimension, and there is no declared default, then the Server **must** throw an Exception to indicate that a value was required. The service exception "code" attribute **should** be set to "MissingDimensionConstraint".

?? *extent_value* declares what value(s) along the Dimension axis are appropriate for this specific Layer, Coverage or Feature. The extent_value has the following syntax:

Syntax	Meaning
Value	A single value
Value1,value2,value3	A list of three values
Min/max/resolution	An interval defined by its lower and upper bounds and the resolution (post spacing)
Min1/max1/res1,min2/max2/res2	A list of two intervals

Here is an example Layer element with Extents specified in WMS Capabilities XML:

```
<Layer>
  <Dimension name="time" units="ISO8601" />
  <Dimension name="temperature" units="Kelvin" unitSymbol="K" />
  <Dimension name="elevation" units="EPSG:4301" />
     …
  <Layer>  …
    <Extent name="time" default="2000-10-17">1996-01-01/2000-10-17/P1D</Extent>
    <Extent name="elevation" default="0">0/10000/100</Extent>
    <Extent name="temperature" default="300">230,temp2,temp3</Extent>
</Layer>
```

…
</Layer>

C.3.2 Multi-Dimensional Bounds And Resolution: Extended C.3.3 BoundingBox

[Do we need extended BBox if we have <Extent name="time|elevation">?]
In the WMS 1.0 Capabilities XML response, the BoundingBox element had attributes minx, miny, maxx, maxy to describe a Layer's extent as a rectangle within a 2-D Spatial Reference System (SRS). This document generalizes the BoundingBox element when a Layer's domain is expressed according to a 3-D or 4-D reference system

1. New optional attributes minz, maxz, mint, and maxt describe a 3-D box or 4-D hypercube within a 3-D or 4-D coordinate reference system
2. New optional attributes resx, resy, resz, rest indicate the spatio-temporal resolution of data within the 2-, 3-, or 4-dimensional spatio-temporal reference system.

Here are a few examples:

2-D Bounding Box:
```
<BoundingBox SRS="EPSG:26918"
minx="341989" miny="4798657" maxx="456436" maxy="4987564"
resx="1" resy="1">
```
3-D Bounding Box:
```
<BoundingBox CRS="ECR01"
minx="341989" miny="4798657" maxx="456436" maxy="4987564"
minz="1000" maxz="100000" resx="1" resy="1" resz="1000">
```
4-D Bounding Box:
```
<BoundingBox CRS="ST00054"
minx="341989" miny="4798657" maxx="456436" maxy="4987564"
minz="1000" maxz="100000" mint="1999-08-01" maxt="2000-08-01"
resx="1" resy="1" resz="1000" rest="P1D">
```

C.3.4 Bounding Geometry

Finally, a layer may specify its spatio-temporal extent more precisely, using a GML geometry encoded in a BoundingGeometry. This geometry may be 2-, 3-, or 4-dimensional.

C.4 GetMap and GetCoverage Requests

Requests against multi-dimensional data Layers described as above requires a few simple extensions to the GetMap protocol of WMS, in order to express query constraints on a Layer's spatio-temporal domain and its multi-dimensional range. The following sections explain these query constraints, and present a few examples.

C.4.1 Query constraints on elevation and time

Query constraints on elevation and time require a different syntax depending on whether the queried Layer encodes its domain's elevation and time as distinct Extents or only as a 3-D or 4-D BoundingBox.

Note: A GetMap request requires a single value (rather than a vector) at each 2-D location, so it must request a single-value "slice" of available data along the Time and Elevation axes. GetCoverage lifts this restriction by allowing multi-dimensional values at each 2-D location.

C.4.1.1 Elevation and time expressed as distinct Extents

If a Layer has an elevation extent defined, then GetCoverage and GetMap requests may supplement a BBOX constraint with

ELEVATION=*value*

If a Layer has a time extent defined, then requests may use

TIME=*value*

In each case, *value* may be a scalar value, a comma-separated list, or an interval, as described in C.3.1 above. The absence of either of these parameters is equivalent to a request for the Layer's default value (if defined) along that dimension. This is a special case of the dimensional constraints described below in C.4.2.

C.4.1.2 Elevation and time embedded in a 3-D or 4-D BoundingBox

When a Layer's domain is expressed (only) as a 3-D or 4-D BoundingBox, queries must use an extended BBOX syntax to express a query constraint as a 3-D bounding box or a 4-D hypercube:

BBOX=minx,miny,maxx,maxy,*minz,maxz,mint,maxt*

In the above syntax, *minz,maxz* and *mint,maxt* are optional terms. By omitting the *minz,maxz* or *mint,maxt* terms, a query requests the Layer's default value (if defined) along the 3rd spatial dimension (*z*) or the time dimension (*t*). (To specify *t* extents but omit *z* extents, clients should use null *z* values, *viz.* BBOX=minx,miny,maxx,maxy,,,mint,maxt.)

As in WMS 1.0, the 2-D rectangle (minx,miny,maxx,maxy) is always required. (For simplicity and WMS 1.0 compatibility in the 2-D case, the order of these four numbers differs from that of the optional z and time numbers.)

C.4.2 Query constraints against sample dimensions

The Capabilities XML response describes the extent of each Layer along a set of sample dimensions, and gives each Dimension a unique name (<Dimension name="___" …>). Subsequent GetMap or GetCoverage requests may use each these dimension names as a query constraint parameter. The value of such parameter would denote subsets of the Layer's Extent along that dimension, using the syntax described in C.3.1 above. Queries may omit dimension names (or use nulls), to request the default value along those dimensions. (If no default value exists along that dimension, the server must throw an exception, as specified in C.3.1.)

For instance, if a Layer is described as having an extent along a Dimension named "wavelength", then requests against that Layer may include the constraint "WAVELENGTH=…". Further examples appear next.

Note: A GetCoverage query may request a coverage with multiple sample dimensions. However, a GetMap query requires a single value (rather than a vector) at each 2-D location, so it must request a single-value "slice" of available data along all but one of the Layer's sample dimensions.

C.4.3 Example requests

The following are examples of multi-dimensional GetMap and GetCoverage requests.

GetMap (single ozone Map at specified time and height):

VERSION=x.y.z	WIDTH=600
REQUEST=map	HEIGHT=300
LAYERS=ozone	TIME=2000-08-03
SRS=EPSG:4326	ELEVATION=1000
BBOX=-180,-90,180,90	FORMAT=PNG

GetMap (movie loop at specified frame times):

VERSION=x.y.z	WIDTH=600

REQUEST=map	HEIGHT=300
LAYERS=ozone	TIME=2000-07-01/2000-07-31/P1D
SRS=EPSG:4326	ELEVATION=1000
BBOX=-180,-90,180,90	FORMAT=MPEG

GetCoverage (4D block of data with dimension x,y,z,t):

VERSION=x.y.z	SKIPX=10
REQUEST=coverage	SKIPY=10
LAYERS=layer	TIME=2000-07-01/2000-07-31/P1D
SRS=srs_identifier	ELEVATION=0/10000/1000
BBOX=minx,miny,maxx,maxy	FORMAT=HDFEOS

APPENDIX D: MULTI-DIMENSIONAL COORDINATE REFERENCE SYSTEMS (EXPERIMENTAL)

D.1 Introduction

This section specifies a standard model for data defining Coordinate Reference Systems (CRSs) as given in "OGC Specification Recommended Definition Data for Coordinate Reference Systems and Coordinate Transformations - Draft (OGC Document 01-014r1). The specification provide here lacks the descriptive comments provided in 01-014r1.

Most of the geospatial concepts used in this section are from the current OGC abstract model of Spatial Reference Systems (Topic 2, document 01-102). The OGC Spatial Reference Systems concepts are largely consistent with ISO/TC 211 DIS 19111 (N 934), although different terms are sometimes used for the same concepts.

This clause specifies an XML DTD for CRS definition data in the following parts:

1. Coordinate Reference System definition
2. Datum definitions
3. Ellipsoid and Prime Meridian definitions

These XML DTDs are specified as external DTDs.

D.2 XML for coordinate reference system definition

This subclause presents the proposed XML DTD package for transfer of a Coordinate Reference System definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT CoordinateReferenceSystem (
        NameSet?,
        (Identifier
        | (ValidityRegion?,
          (  CompoundCRS
          |  GeocentricCRS
          |  Geographic3dCRS
          |  Geographic2dCRS
          |  ProjectedCRS
          |  LocalCRS
          |  ImageCRS
          |  VerticalCRS
          |  TemporalCRS  ) ) ) >
<!ELEMENT CompoundCRS (
        CoordinateReferenceSystem+) >
```

```
<!ELEMENT GeocentricCRS (
        GeodeticDatum,
        CartesianCoordinateSystem) >
<!ELEMENT Geographic3dCRS (
        GeodeticDatum,
        EllipsoidalCoordinateSystem) >
<!ELEMENT Geographic2dCRS (
        GeodeticDatum,
        EllipsoidalCoordinateSystem) >
<!ELEMENT ProjectedCRS (
        CartesianCoordinateSystem,
        CoordinateReferenceSystem,
        CoordinateTransformationDefinition) >
<!ELEMENT LocalCRS (
        (    CartesianCoordinateSystem
        | ObliqueCartesianCoordinateSystem
        | EllipsoidalCoordinateSystem
        | GravityRelatedHeightCS
        | PolarCoordinateSystem
        | CylindricalCoordinateSystem
        | LinearCoordinateSystem
        | UserDefinedCoordinateSystem),
        (LocalDatum
          | (CoordinateReferenceSystem,
          CoordinateTransformationDefinition) ) ) >
<!ELEMENT ImageCRS (
        ( (CartesianCoordinateSystem
        | ObliqueCartesianCoordinateSystem ),
        AxesOrigin )
        | ( ImageDatum
        | (CoordinateReferenceSystem,
        CoordinateTransformationDefinition ) ) ) >
<!ELEMENT AxesOrigin   (#PCDATA) >
<!ELEMENT VerticalCRS (
        GravityRelatedCS,
        VerticalDatum) >
<!ELEMENT TemporalCRS (
        temporalCS,
        origin) >
<!ELEMENT origin   (#PCDATA) >
<!-- End of XML DTD for Coordinate Reference Systems -->
```

D.3 XML for coordinate system definitions

```
<!-- Version 0.0 of XML DTD for Coordinate System definition. This DTD uses XML elements
that are specified in other DTDs. -->
<!ELEMENT CartesianCoordinateSystem (
        NameSet?,
        (Identifier
        | (dimensions,
          CoordinateAxis+,
          Identifier?) ) ) >
<!ELEMENT dimensions   (#PCDATA) >
<!ELEMENT ObliqueCartesianCoordinateSystem (
        NameSet?,
        (Identifier
        | (dimensions,
          CoordinateAxis+,
          Identifier?) ) ) >
<!ELEMENT EllipsoidalCoordinateSystem (
        NameSet?,
        (Identifier
        | (dimensions,
          CoordinateAxis+,
          Identifier?) ) ) >
<!ELEMENT GravityRelatedHeightCS (
        NameSet?,
        (Identifier
        | (dimensions,
          CoordinateAxis,
          Identifier?) ) ) >
<!ELEMENT PolarCoordinateSystem (
        NameSet?,
        (Identifier
```

```
        | (dimensions,
          CoordinateAxis+,
          Identifier?) ) ) >
<!ELEMENT CylindricalCoordinateSystem (
        NameSet?,
        (Identifier
        | (dimensions,
          CoordinateAxis+,
          Identifier?) ) ) >
<!ELEMENT LinearCoordinateSystem (
        NameSet?,
        (Identifier
        | (dimensions,
          CoordinateAxis,
          Identifier?) ) ) >
<!ELEMENT TemporalCoordinateSystem (
        NameSet?,
        (Identifier
        | (dimensions,
          CoordinateAxis,
          Identifier?) ) ) >
<!ELEMENT UserDefinedCoordinateSystem (
        NameSet?,
        (Identifier
        | (dimensions,
          CoordinateAxis+,
          Identifier?) ) ) >
<!ELEMENT CoordinateAxis (
        NameSet?,
        (Identifier
        | (axisAbbreviation?,
        axisDirection,
        (LinearUnit | AngularUnit | TimeUnit |
         PixelSpacingUnit),
        Identifier?) ) ) >
<!ELEMENT axisAbbreviation   (#PCDATA) >
<!ELEMENT axisDirection      (#PCDATA) >
<!-- End of XML DTD for Coordinate Systems -->
```

D.4 XML for datum definitions

This subclause presents the XML DTD package for transfer of datum definitions.

```
<!-- Version 0.0 of XML DTD for Datum definitions. This DTD uses XML elements that are
specified in other DTDs. -->
<!ELEMENT VerticalDatum (
        NameSet?,
        (Identifier
        | (datumType,
           Identifier?) ) ) >
<!ELEMENT datumType   (#PCDATA) >
<!ELEMENT LocalDatum (
        NameSet?, Identifier ) >
<!ELEMENT ImageDatum (
        NameSet?, Identifier ) >
<!ELEMENT GeodeticDatum (
        NameSet?,
        (Identifier
        | (Ellipsoid,
           PrimeMeridian?,
           Identifier?) ) ) >
<!-- End of XML DTD for Datum definitions -->
```

D.5 XML for ellipsoid and prime meridian definitions

This subclause presents the proposed XML DTD package for transfer of Ellipsoid and Prime Meridian definitions.

```
<!-- Version 0.0 of XML DTD for Ellipsoid and Prime Meridian definitions. This DTD uses XML
elements that are specified in other DTDs. -->
<!ELEMENT Ellipsoid (
        NameSet?,
        (Identifier
        | (LinearUnit,
          semiMajorAxis,
          semiMinorAxis,
          inverseFlattening,
          Identifier?) ) ) >
    <!ATTLIST Ellipsoid
        flatteningDefinitive   (true | false)   #REQUIRED
        ellipsoidShape         (true | false)   #IMPLIED >
<!ELEMENT semiMajorAxis       (#PCDATA) >
<!ELEMENT semiMinorAxis       (#PCDATA) >
<!ELEMENT inverseFlattening   (#PCDATA) >
<!ELEMENT PrimeMeridian (
        NameSet?,
        (Identifier
        | (greenwichLongitude,
          AngularUnit,
          Identifier?) ) ) >
<!ELEMENT greenwichLongitude   (#PCDATA) >
<!-- End of XML DTD for Ellipsoid and Prime Meridian -->
```

D.6 Integration into Capabilities schema

The Layer encoding shall allow either a CRS or an SRS element. That is, the DTD describing the Layer element changes from

<!ELEMENT Layer (Name?, Title, …, SRS?, …) >

to

<!ELEMENT Layer (Name?, Title, …, (SRS|CRS)? …) >

D.7 CRS Examples

Example XML for well-known CRS

This subclause provides example XML using the Coordinate Reference System definition XML element containing only the Identifier data element. This example XML is applicable to a well-known CRS included in XML formatted feature data,.

```
<CoordinateReferenceSystem>
    <Identifier>
        <code>4326</code>
        <codeSpace>EPSG</codeSpace>
    </Identifier>
</CoordinateReferenceSystem>
```

Example XML for compound coordinate system

This subclause provides a simplified example XML for a 3D compound coordinate system, that omits most of the optional elements and attributes. This simplified example XML is intended to be adequate when the Projected Coordinate System and the Vertical Coordinate System are both well-known, as input to server software implementing the high-level Coordinate Transformation interface. That is, this simplified example assumes that the server software can obtain the CoordinateSystemDefinition and other data for these two coordinate systems using only the Identifier that is input. This simplified example XML also assumes the coordinate axes from the two compounded coordinate systems are sequentially combined.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE CoordinateReferenceSystem SYSTEM
       "http://opengis.org/xmldtds/transformations.dtd">
<CoordinateReferenceSystem>
    <CoordinateSystemDefinition>
        <dimensions>3</dimensions>
        <CoordinateAxis>
            <axisName>E</axisName>
            <axisDirection>east</axisDirection>
            <LinearUnit>
                <Identifier>
                    <code>9001</code>
                    <codeSpace>EPSG</codeSpace>
                </Identifier>
            </LinearUnit>
        </CoordinateAxis>
        <CoordinateAxis>
            <axisName>N</axisName>
            <axisDirection>north</axisDirection>
            <LinearUnit>
                <Identifier>
                    <code>9001</code>
                    <codeSpace>EPSG</codeSpace>
                </Identifier>
            </LinearUnit>
        </CoordinateAxis>
        <CoordinateAxis>
            <axisName>U</axisName>
            <axisDirection>up</axisDirection>
            <LinearUnit>
                <Identifier>
                    <code>9001</code>
                    <codeSpace>EPSG</codeSpace>
                </Identifier>
            </LinearUnit>
        </CoordinateAxis>
    </CoordinateSystemDefinition>
    <CompoundCoordinateSystem>
        <CoordinateReferenceSystem>
            <Identifier>
                <code>27700</code>
                <codeSpace>EPSG</codeSpace>
            </Identifier>
        </CoordinateReferenceSystem>
        <CoordinateReferenceSystem>
            <Identifier>
                <code>5701</code>
                <codeSpace>EPSG</codeSpace>
            </Identifier>
        </CoordinateReferenceSystem>
    </CompoundCoordinateSystem>
    <Identifier>
        <code>7405</code>
        <codeSpace>EPSG</codeSpace>
    </Identifier>
</CoordinateReferenceSystem>
```