Open GIS Consortium Inc.

Date: 2002-04-04

Reference number of this OpenGIS© Project Document: **OGC 02-024**

Version: 0.7

Category: OpenGIS© OGC Interoperability Program Report - Engineering Specification

Editor: John D. Evans

OWS1 Web Coverage Service (WCS)

Copyright notice

Warning

This document is not an OGC Standard or Specification. This document presents a discussion of technology issues considered in an Interoperability Initiative of the OGC Interoperability Program. The content of this document is presented to create discussion in the geospatial information industry on this topic; the content of this document is not to be considered an adopted specification of any kind. This document does not represent the official position of the OGC nor of the OGC Technical Committee. It is subject to change without notice and may not be referred to as an OGC Standard or Specification. However, the discussions in this document could very well lead to the definition of an OGC Implementation Specification.

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© Interoperability Program Report-Engineering Specification
Document subtype: if applicable
Document stage: Final
Document language: English

Contents

i. Preface

ii. Submitting organizations

This Interoperability Program Report – Engineering Specification is being submitted to the OGC Interoperability Program by the following organizations:

iii. Document Contributor Contact Points

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

CONTACT	COMPANY	ADDRESS	PHONE/ FAX	EMAIL
John D. Evans	GST, Inc.	6411 Ivy Ln. Ste 300 Greenbelt, MD 20770 USA	(301) 474-9696	evans@gst.com
Stephane Fellah	PCI Geomatics	490 St. Joseph Blvd., Suite 400 Hull, Quebec J8Y 3Y7 CANADA	(819) 770-0022 Ext. 223	fellah@pcigeomatics.com
Jeff Lansing	Polexis, Inc.			jeff@polexis.com

iv. Revision history

Date	Release	Author	Paragraph modified	Description
2001-11-17	0.5	John Evans		Initial DIPR version
2001-11-29	0.5	Jeff Lansing		Added DescribeCoverageType content.
2001-11-29	0.5	Stephane Fellah		Revised Format and Interpolation definitions; many substantive comments
2002-01-31	0.5.1	John Evans		Revisions & corrections; XML Schema
2002-02-20	0.6	Stephane Fellah		New schema for coverage layer descriptions and XML requests
2002-04-04	0.7	John Evans		Fixed and streamlined the 0.6 schema; added compound observations; documentation and integration.

v. Changes to the OpenGIS® Abstract Specification

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

Foreword

OGC 01-018r3 consists of the following part: Web Coverage Service Report.

Introduction

The Web Coverage Service (WCS) supports the networked interchange of geospatial data as "coverages" containing values or properties of geographic locations. Unlike the Web Map Service (WMS) (OGC document #01-021r2), which filters and portrays spatial data to return static maps (server-rendered as pictures), the Web Coverage Service provides access to intact (unrendered) geospatial information, as needed for client-side rendering, multi-valued coverages, and input into scientific models and other clients beyond simple viewers.

The Web Coverage Service consists of three operations: *GetCapabilities*, *GetCoverage, and DescribeCoverageType*. The *GetCapabilities* operation returns an XML document describing the service and the data collections from which clients may request coverages. Clients would generally run the *GetCapabilities* operation and cache its result for use throughout a session, or reuse it for multiple sessions.

The *GetCoverage* operation of a Web Coverage Service is normally run after *GetCapabilities* has determined what queries are allowed and what data are available. The *GetCoverage* operation returns values or properties of geographic locations, bundled in a well-known coverage format. Its syntax and semantics are similar to the WMS *GetMap* request, but several extensions support the retrieval of coverages rather than static maps.

OWS1 Web Coverage Service (WCS)

1 Scope

This specification document explains how WCS serves to describe, request, and deliver multi-dimensional coverage data over the World Wide Web. This version of the Web Coverage Service emphasizes "simple" coverages (defined on some regular, rectangular grid or tesselation of space); and anticipates other coverage types defined in the OpenGIS Abstract Specification (Topic 6, "The Coverage Type," OGC document #99-106).

2 Conformance

Conformance and Interoperability Testing for this OGC Interoperability Program Report may be checked using all the relevant tests specified in Annex A (normative). The framework, concepts, and methodology for testing, and the criteria to be achieved to claim conformance, are specified in ISO 19105: Geographic information — Conformance and Testing.

3 Normative references

The following normative documents contain provisions that, through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of OGC nnn are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies.

OGC 01-021r2:2001, *Web Map Service v. 1.1*

ISO 19123, *Geographic Information — Coverage Geometry and Functions.*

Abstract Specification Topic 0: Overview, OGC document 99-100r1

Guidelines for Successful OGC Interface Specifications, OGC document 00-014r1

4 Terms and definitions

For the purposes of this document, the terms and definitions given in the above references apply, as do the following terms.

operation
specification of a transformation or query that an object may be called to execute [OGC AS 12]

interface
named set of **operations** that characterize the behavior of an entity [OGC AS 12]

service
distinct part of the functionality that is provided by an entity through **interfaces** [OGC AS 12]

service instance
server
actual implementation of a **service**

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

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

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

map
pictorial representation of geographic data

capabilities XML
service-level metadata describing the **operations** and content available at a **service instance**.

5 Conventions

5.1.1 Symbols (and abbreviated terms)

The following symbols and abbreviated terms are used in this document.

API	Application Program Interface
DCP	Distributed Computing Platform

ISO	International Organization for Standardization
OGC	Open GIS Consortium
UML	Unified Modeling Language
XML	eXtensible Markup Language
1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional

5.1.2 UML Notation

The diagrams that appear 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.

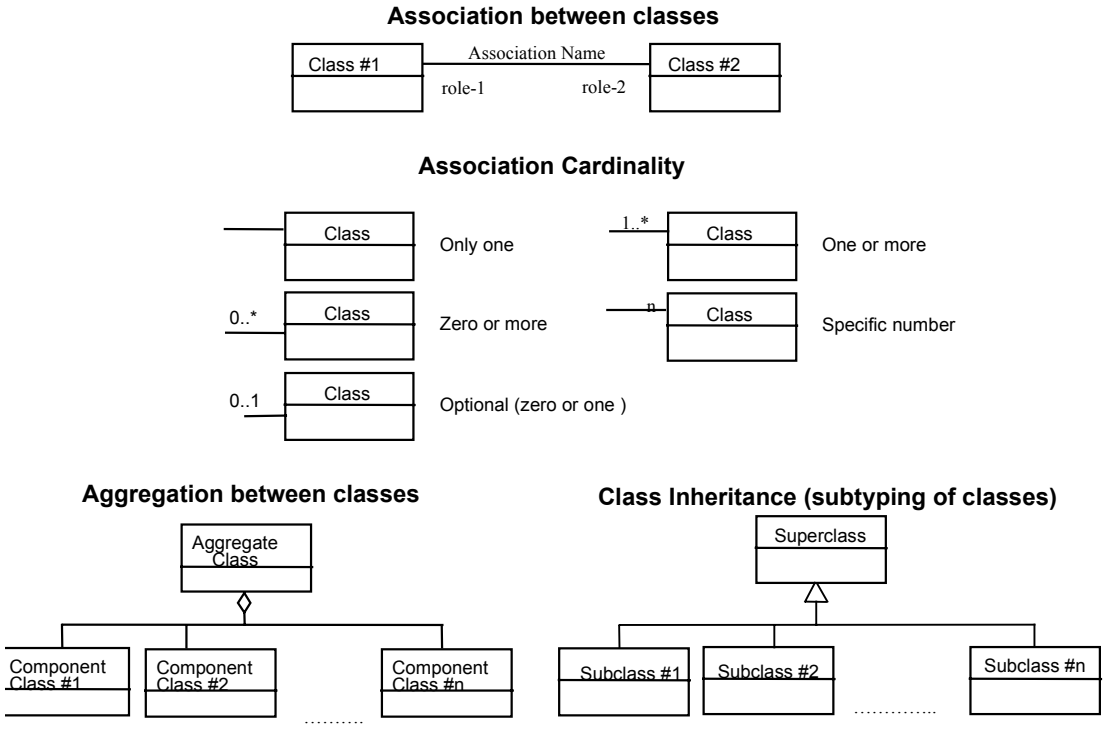

Figure 1 - UML Notation

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

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

b) <<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.

c) <<CodeList>> is a flexible enumeration that uses string values for expressing a list of potential values.

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

a) CharacterString – A sequence of characters

b) Integer – An integer number

c) Double – A double precision floating point number

d) Float – A single precision floating point number

6 Basic Service Elements

This section describes aspects of Web Coverage Server behavior (more generally, of OGC Web Service behavior) that are independent of particular operations or are common to several operations or interfaces.

6.1 Version Numbering and Negotiation

6.1.1 Version Number Form

The published specification version number contains three positive integers, separated by decimal points, in the form "x.y.z". The numbers "y" and "z" will never exceed 99. Each OWS specification is numbered independently.

6.1.2 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.

6.1.3 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.

6.1.4 Version Number Negotiation

An OWS Client may negotiate with a Service Instance to determine a mutually agreeable specification version. Negotiation is performed using the GetCapabilities operation [see Section 7 below] according to the following rules.

All Capabilities XML must include a protocol version number. 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.

Version number negotiation occurs as follows:

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

2. If a version unknown to the server is requested, the server **must** send the highest version less than the requested version.

3. 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.

4. 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).

5. 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 1: 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 2: 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.

6.2 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)[6]. Thus the Online Resource of

each operation supported by a service instance is an HTTP Uniform Resource Locator (URL). The URL may be different for each operation, or the same, at the discretion of the service provider. Each URL **must** conform to the description in [6] 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 methods: GET and POST. One or both of these methods may be defined for a particular OGC Web Service type and offered by a service instance, and the use of the Online Resource URL differs in each case.

An Online Resource URL intended for HTTP GET requests is in fact only a URL prefix to which additional parameters must be appended in order to construct a valid Operation request. A URL prefix is defined as an opaque string including the protocol, hostname, optional port number, path, a question mark '?', and, **optionally**, one or more server-specific parameters ending in an ampersand '&'. The prefix uniquely identifies the particular service instance. For HTTP GET, the URL prefix **must** end in either a '?' (in the absence of additional server-specific parameters) or a '&'. In practice, however, Clients **should** be prepared to add a necessary trailing '?' or '&' before appending the Operation parameters defined in this specification in order to construct a valid request URL.

An Online Resource URL intended for HTTP POST requests is a complete and valid URL to which Clients transmit encoded requests in the body of the POST document. A WCS server **must not** require additional parameters to be appended to the URL in order to construct a valid target for the Operation request.

6.2.1 Key-Value Pair encoding

Using Key-Value Pair encoding, a client composes the necessary request parameters as keyword/value pairs in the form "keyword=value". These may be transmitted to the server via HTTP GET or HTTP POST.

For HTTP GET, the client appends these keyword/value pairs, separated by '&', to the Online Resource URL prefix for the chosen Operation. The resulting URL **must** be valid according to the HTTP Common Gateway Interface (CGI) standard [7], which mandates the presence of '?' before the sequence of query parameters and the '&' between each parameter. As with all CGI applications, the query URL **must** be encoded [8] to protect special characters.

Table 1 summarizes the components of an operation request URL for HTTP GET.

Table 1 – A general OGC Web Service Request

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. The prefix is

	entirely at the discretion of the service provider.
Name=value&	One or more standard request parameter name/value pairs defined by an OGC Web Service. The actual list of required and optional parameters is mandated for each operation by the appropriate OWS specification.

A request encoded as key-value pairs using the HTTP GET method may be saved as a bookmark, embedded as a hyperlink, or referenced via Xlink in an XML document.

6.2.1.1 Parameter Ordering and Case

Parameter names **shall not** be case sensitive, but parameter values **shall** be case sensitive. In this document, parameter names are typically shown in uppercase for typographical clarity, not as a requirement.

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.

6.2.1.2 Parameter Lists

Parameters consisting of lists shall use the comma (",") as the delimiter between items in the list: e.g., `parameter=item1,item2,item3.` Multiple lists can be specified as the value of a parameter by enclosing each list in parentheses ("(", ")"): e.g., `parameter=(item1a,item1b,item1c)(item2a,item2b,item2c.`

6.2.2 XML encoding

[Debate about whether to use SOAP[19] has not been completely settled by tests on actual implementations, so two alternatives are temporarily described here.]

Clients may also encode requests in XML for transmission to the server using HTTP GET or (more often) HTTP POST. The XML request must conform to the schema corresponding to the chosen operation, and the client must send it to the URL listed for that operation in the server's Capabilities XML file.

To support SOAP messaging, clients need only enclose this XML document in a SOAP envelope as follows:

```
<env:Envelope
 xmlns:env="http://www.w3.org/2001/09/soap-envelope">
  <env:Body>
    request document here
  </env:Body>
</env:Envelope>
```

The client must precede the XML request with the appropriate HTTP request headers defined below (along with any others mandated by [8]):

- `Content-Type: text/xml; charset="utf-8"`
  `-or?-`
  `Content-Type: application/vnd.ogc.`*`request`*`+xml`
  (where *`request`* is one of the defined operation names)
- `Content-Length: nnnn`
  (Where `nnnn` is the number of bytes in the request document.

SOAP messaging requires the following headings:
- `Content-Type: text/xml; charset="utf-8"`
- `Content-Length: nnnn`
  where `nnnn` is the number of bytes in the document, including the SOAP envelope
- `SOAPAction: {URL} (may be null)`

6.3 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 in Section 6.4 below.

NOTE: 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.

The appropriate Multipurpose Internet Mail Extensions (MIME) type [9] must accompany response objects.

SOAP requests return a multipart MIME document formatted according to the SOAP Messages with Attachments [20] note from W3C. The first part is a SOAP message; the second part contains the actual server response (image, coverage, etc.). Alternatively, the response may be a SOAP message that includes the URL of the actual response from the server.

Other HTTP entity headers should accompany response objects 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.

6.4 Service Exceptions

Upon receiving a request that 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. For example: in the HTTP DCP, if the URL prefix is incorrect an HTTP 404 status code is sent.

Upon receiving an invalid request, the service **must** issue a Service Exception XML message to describe to the client application or its human user the reason(s) that the request is invalid.

Service Exception XML **must** be valid according to the Service Exception DTD in Annex A.3. In an HTTP environment, the MIME type of the returned XML **must** be "`application/vnd.ogc.se_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 Annex A.4.

Service Exceptions **may** include exception codes as indicated in Annex A.3. Servers **shall not** use these codes for meanings other than those specified. Clients **may** use these codes to automate responses to Service Exceptions.

7 GetCapabilities operation (required)

The Web coverage server must describe its capabilities. This section defines an XML document structure intended to convey general information about the service itself, and specific information about the available data collections from which coverages may be requested.

7.1 GetCapabilities request

7.1.1 Key-Value pair encoding

The general form of a **GetCapabilities** request is defined in the Basic Service Elements section, and summarized in Table 2 below.

Table 2 — The parameters of a GetCapabilities request URL

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

When making this request of a WCS server, which may offer other OGC Web Services as well, it is necessary to indicate that the client seeks information about the WCS server in particular. Thus, the SERVICE parameter of the request **must** have the value "WCS" as shown in Table 2.

7.1.2 XML encoding

The OWS Service Metadata XML IPR presents the XML Schema for the XML encoding of the **GetCapabilities** request. For example, a client may encode a **GetCapabilities** request in XML as follows:

```
<GetCapabilities version="0.7" service="WCS">
  <Section>/OGC_Capabilities/OperationSignatures</Section>
</GetCapabilities>
```

The top-level XML element, **GetCapabilities**, has two attributes, **version** and **service**, denoting, respectively, the version number of the protocol and the service it addresses.

The **GetCapabilities** element has one optional sub-element, **Section**, denoting which portion of the Capabilities XML document to return: the **ServiceOffering**, **OperationSignatures**, or **ContentMetadata** elements. If this element is not supplied, the server must return the entire Capabilities XML document.

7.2 GetCapabilities response: Capabilities XML document

The response to a **GetCapabilities** request is a Capabilities XML document conforming to the Schema given in the OWS Service Metadata XML IPR, composed of four main sections depicted in Figure 2:

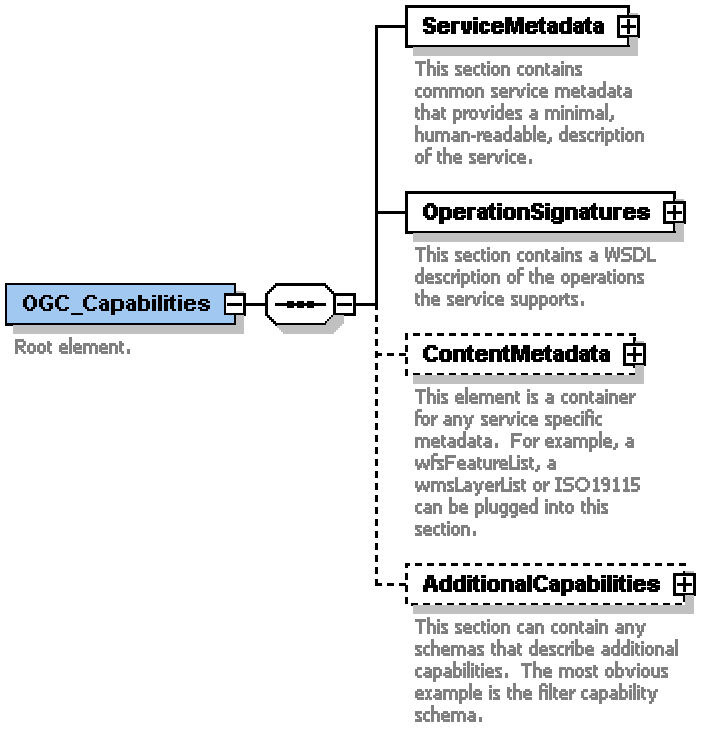

Figure 2 - OGC_Capabilities top-level element

The OWS Service Metadata XML IPR describes the role of these four sections, and the structure of the first two (**ServiceMetadata** and **OperationSignatures**).

OperationSignatures is a WSDL sevice description that imports several schemas, to define the syntax of requests (key-value pairs, XML, SOAP) and responses (XML documents, exceptions, or coverage data). These schemas are detailed below in sections ___ and __.

The third section of the Capabilities XML file (**ContentMetadata**) is specific to WCS, and imports the **CoverageLayerList** schema, described next.

7.2.1 CoverageLayerList top-level element

A the top level of the **ContentMetadata** section of the Capabilities XML document for WCS is a **CoverageLayerList** element, which lists coverage layers in any order. Figure 3 shows the structure of the CoverageLayerList element.

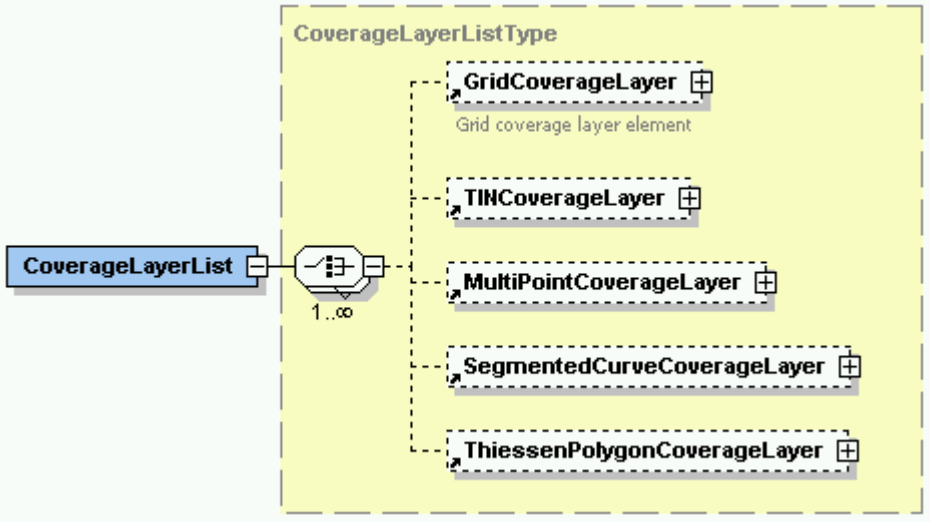

Figure 3 - Coverage layer list (top level under ContentMetadata)

The coverage layers are of 5 different kinds, but they all share several common properties, listed next.

7.2.2 CoverageLayer properties (common)

In its Capabilities XML response, a Web Coverage Server announces that it can return coverages (that is, values or properties of spatio-temporal locations) culled from various collections of data. Each logical collection from which coverages may be requested is called a coverage layer. The different kinds of coverage layers available all share several common elements, which comprise the AbstractCoverageLayerType (Figure 4).

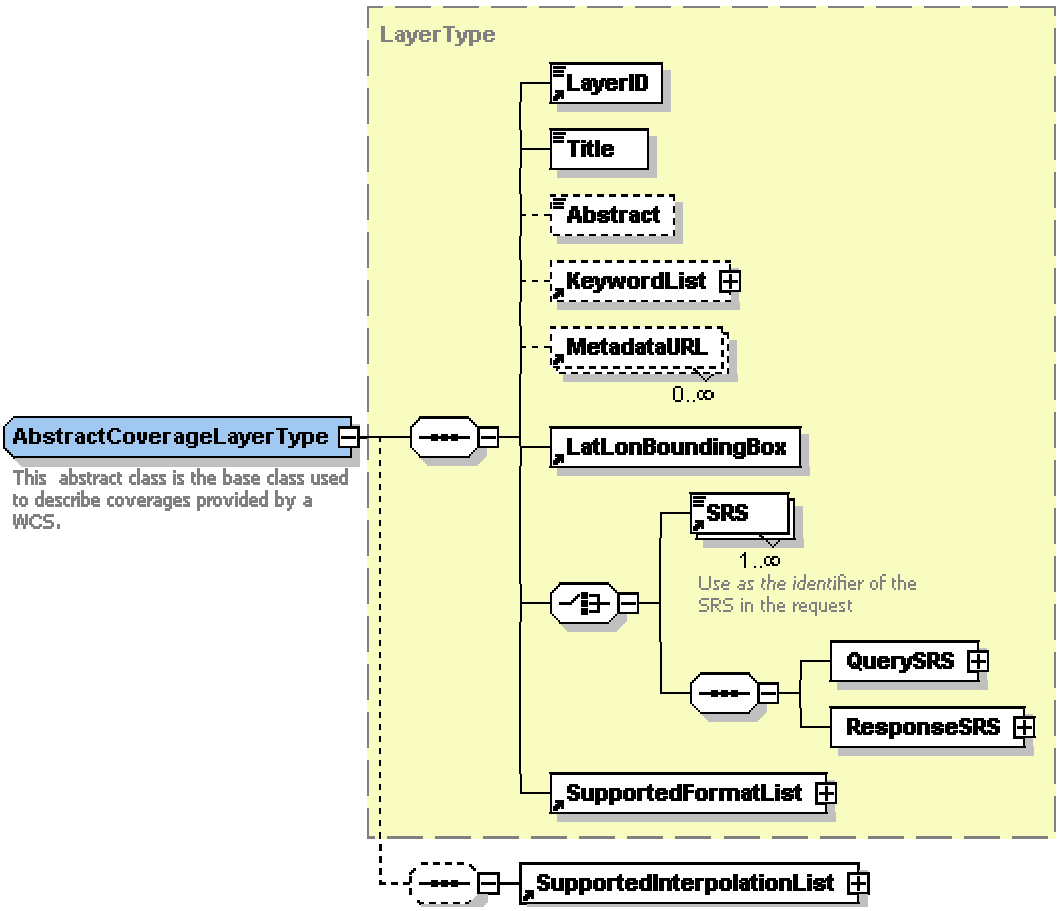

Figure 4 - AbstractCoverageLayerType

(**AbstractCoverageLayerType** itself extends the generic **LayerType**, which is meant to be shared with the Web Map Service and Web Feature Service.)

The following sections describe each sub-element of the abstract coverage layer type – these are shared by all the different kinds of coverage layers.

7.2.2.1 LayerID

The *required* **LayerID** is a unique identifier (not used for any other coverage layer). Using the **GetCoverage** or **DescribeCoverage** operation, clients request coverages from that Coverage Layer (or a description of the Coverage Layer) by using this **LayerID** value in the **LAYERS** parameter (for requests expressed as key-value pairs) or in the **LayerID** element (for XML requests).

7.2.2.2 Title

The *required* **Title** element contains a human-readable string for presentation in a menu.

7.2.2.3 **Abstract** and **KeywordList**

The *optional* **Abstract** and **KeywordList** elements are *recommended*. **Abstract** is a narrative description of the map layer. **KeywordList** contains keywords to aid in catalog searches.

7.2.2.4 **MetadataURL**

The *optional* **MetadataURL** element is *recommended* for access to detailed, standardized metadata about the data underneath a particular layer. The **type** attribute indicates the standard to which the metadata complies. Two types are defined at present: *'TC211'* = ISO TC211 19115; *'FGDC'* = FGDC Content Standard for Digital Geospatial Metadata.

7.2.2.5 **LatLonBoundingBox**

The *required* **LatLonBoundingBox** element, has attributes indicating the edges of an enclosing rectangle in longitude/latitude decimal degrees. **LatLonBoundingBox** *must* be supplied regardless of what SRS the map server may support for coverage requests and responses. However, it *may* be approximate if EPSG:4326 (WGS84 geographic) is not a supported request SRS. Its chief purpose is to populate registries for geographic search.

7.2.2.6 **Spatial Reference Systems (SRS): request, response, native**

For each Coverage Layer, the Capabilities XML description specifies (i) the native Spatial Reference System (SRS) of the data; (ii) the SRSs in which it understands incoming GetCoverage requests; and (iii) the SRSs in which it can produce coverages in response to GetCoverage requests.

7.2.2.6.1 **SRS**

Every CoverageLayer *must* have *either* one or more **SRS** elements, *or both* a **QuerySRS** *and* a **ResponseSRS** element (which contain one or more **SRS** elements). A server may choose to detail query and response SRSs separately, or just advertise SRSs in which it can both accept requests and deliver coverage responses.

The content of the **SRS** element may be any of the EPSG: or AUTO: coordinate systems defined in the Web Map Service Implementation Specification; it may be undefined ("**OGC:none**"); or it may be an embedded "swath" referencing system ("**OGC:swath**") that lets the client reconstruct ground coordinates from tie-points or sensor metadata.

7.2.2.6.2 **QuerySRS**

The **QuerySRS** element states the SRS(s) in which GetCoverage requests may be expressed against that coverage layer. One of these *should* be the coverage layer's native SRS.

Requests expressed in the special SRS codes "**OGC:none**" or "**OGC:swath**" refer to subsets defined according the layer's pixel row/column coordinate system (for imagery) or its internal / local coordinate system (for non-gridded data).

7.2.2.6.3 ResponseSRS

The **ResponseSRS** element states the SRS(s) in which coverage replies to GetCoverage requests may be expressed. One of these **should** be the Layer's native SRS. Coverages served in the special SRS codes "**OGC:none**" or "**OGC:swath**" are expressed in pixel row/column coordinate system (for imagery) or an internal / local coordinate system (for non-gridded data). Coverage replies with the SRS "OGC:swath" **may** embed sensor-model or tie-point information to let a thick client georeference the returned coverage.

7.2.2.6.4 NativeSRS

The *optional* **NativeSRS** element states the native SRS of a Coverage Layer. *(This, along with the native resolution stated in the BoundingBox, facilitates client access to unretouched coverage values.).*

7.2.2.7 BoundingBox

Each layer **must** have a BoundingBox element corresponding to its NativeSRS. It **may** also have (or inherit) one for each SRS listed in QuerySRS. The BoundingBox attributes indicate the edges of the Layer's available data in units of the specified SRS.

The attributes resx, resy, resz, rest indicate the spatial resolution of the data in the BoundingBox's SRS. These attributes are recommended for the BoundingBox corresponding to the Native SRS. For a BoundingBox whose SRS is "OGC:none" or "OGC:swath", indicating pixel or local coordinate system, these attributes are usually 1.

7.2.2.8 SupportedFormatList

The *required* **SupportedFormatList** element advertises the output format(s) in which coverages may be requested from this Coverage Layer (e.g., GeoTIFF, HDF-EOS, NITF, DTED, etc.). Both a format name and a MIME type identify these formats. Several OGC-specific types have been defined to distinguish various types of XML documents; these are listed in WMS 1.1.

7.2.2.9 SupportedInterpolationList

The *optional* **SupportedInterpolationList** element states whether and how the server will interpolate a CoverageLayer's values over the spatial domain when a request requires resampling, reprojection, or other generalization. The *optional, repeatable* **InterpolationMethod** sub-element may have one of 8 different values: "nearest neighbor", "linear", "bilinear", "bicubic", "lost area", "barycentric", "piecewise constant", and "none".

SupportedInterpolationList may list an interpolation method in its **default** attribute; this is what clients will get if they don't specify an interpolation method. If **SupportedInterpolationList** has no **default** attribute, and no **InterpolationMethod** sub-element, then clients should assume *nearest-neighbor* interpolation.

If the only Interpolation method listed is 'none', clients may only retrieve coverages from this layer in its native SRS and at its native resolution.

7.2.3 Domain descriptions

The preceding section (7.2.2) describes the **AbstractCoverageLayerType**, whose elements are shared among all of the various kinds of coverage layers. Each actual coverage layer listed in a Capabilities XML document extends this class with two crucial elements, describing the *domain* and the *range* of coverages available from a given coverage layer.

The first of these elements describes the locations in space (and possibly time) for which the coverage layer reports values or measures. These locations are defined in one of five ways:

- a regular grid of pixels or points;
- a triangulated irregular network(TIN);
- a series of points;
- a set of segmented curves;
- a set of Thiessen polygons.

These different kinds of coverages have some elements in common; several other elements are unique to each kind of coverage.

7.2.3.1 Common Elements

The domains of all of the coverage types have a Spatial Extent; some may also have a Temporal Extent, and an Elevation Extent, depicted in Figure 5.

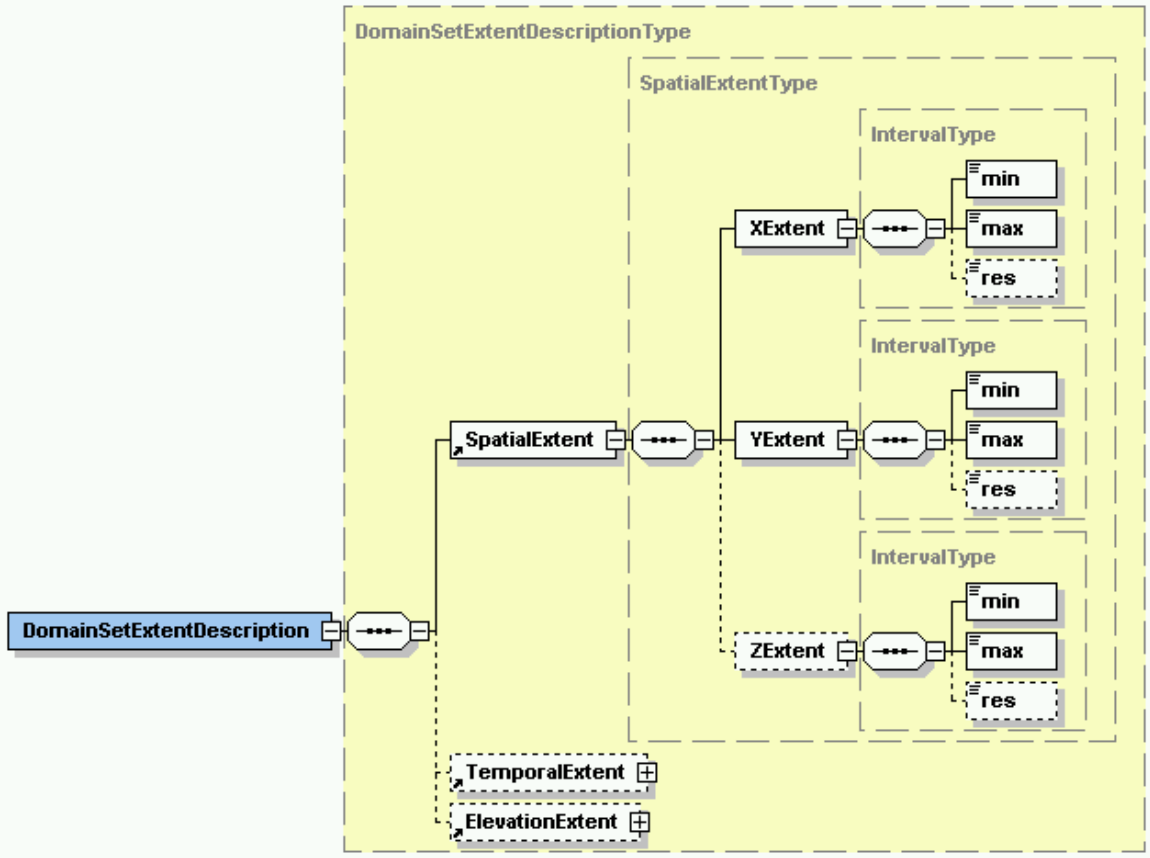

Figure 5 - Domain description common elements: Spatial Extent

- First, a *required* **SpatialExtent** element lists the bounds along each of the spatial dimensions within which coverages may be requested from a coverage layer.

  The **SpatialExtent** element has a *required* attribute **srsName** indicating the spatial reference system in which it is expressed. This may be a 2-D or a 3-D coordinate reference system.

  The *required* **XExtent** and **YExtent** elements define a 2-D spatial extent within a 2-D coordinate reference system. The *optional* **ZExtent** is used for a 3-D spatial extent, expressed in a 3-D coordinate reference system.

- An *optional* **TemporalExtent** element, depicted in Figure 6 below, lists the times for which coverages may be requested from a coverage layer. These may consist of one or more intervals or points in time.

  TemporalExtent has an *optional* attribute **uom** indicating the units of measure in which it expresses time intervals or instances.

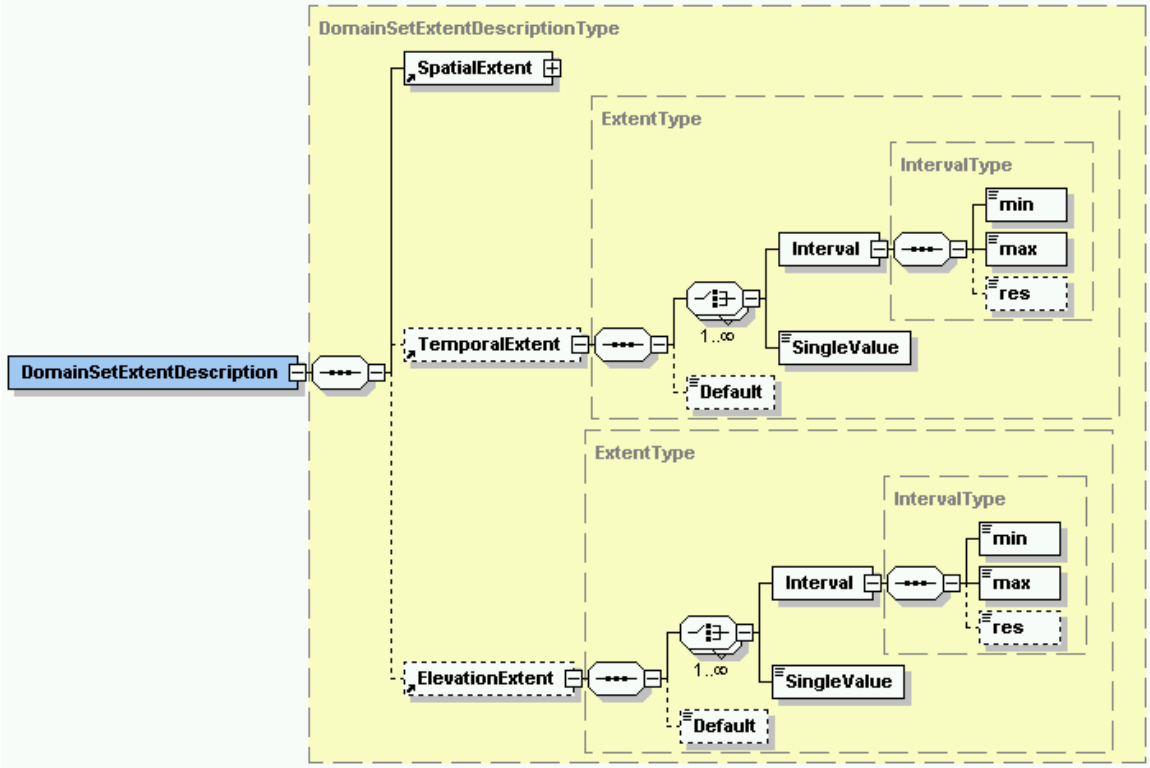

Figure 6 - Domain description common elements: Temporal and Elevation Extent

- Lastly, an *optional* **ElevationExtent** element, also depicted in Figure 6 above, lists the elevation intervals or points at which coverages may be requested from a coverage layer.

  ElevationExtent has an *optional* attribute **uom** indicating the units of measure in which it expresses elevations.

  ElevationExtent is intended to supplement a 2-dimensional spatial extent with elevation information.

Beyond the spatial, temporal, and elevation domain extent, the various kinds of coverage layers add more specific domain descriptors, as detailed in the next several sections.

7.2.3.2 Grid Coverage Layer

First, a grid coverage layer defines its domain as a regular grid of points or cells in 2, 3, or 4 dimensions. This description is suitable for digital airphotos, satellite imagery, gridded terrain models, or any other raster data. As depicted in Figure 7, the Grid Coverage Layer's **DomainSetDescription** has a repeatable element **GridExtentDescription** that extends the shared **DomainSetExtentDescriptionType**.

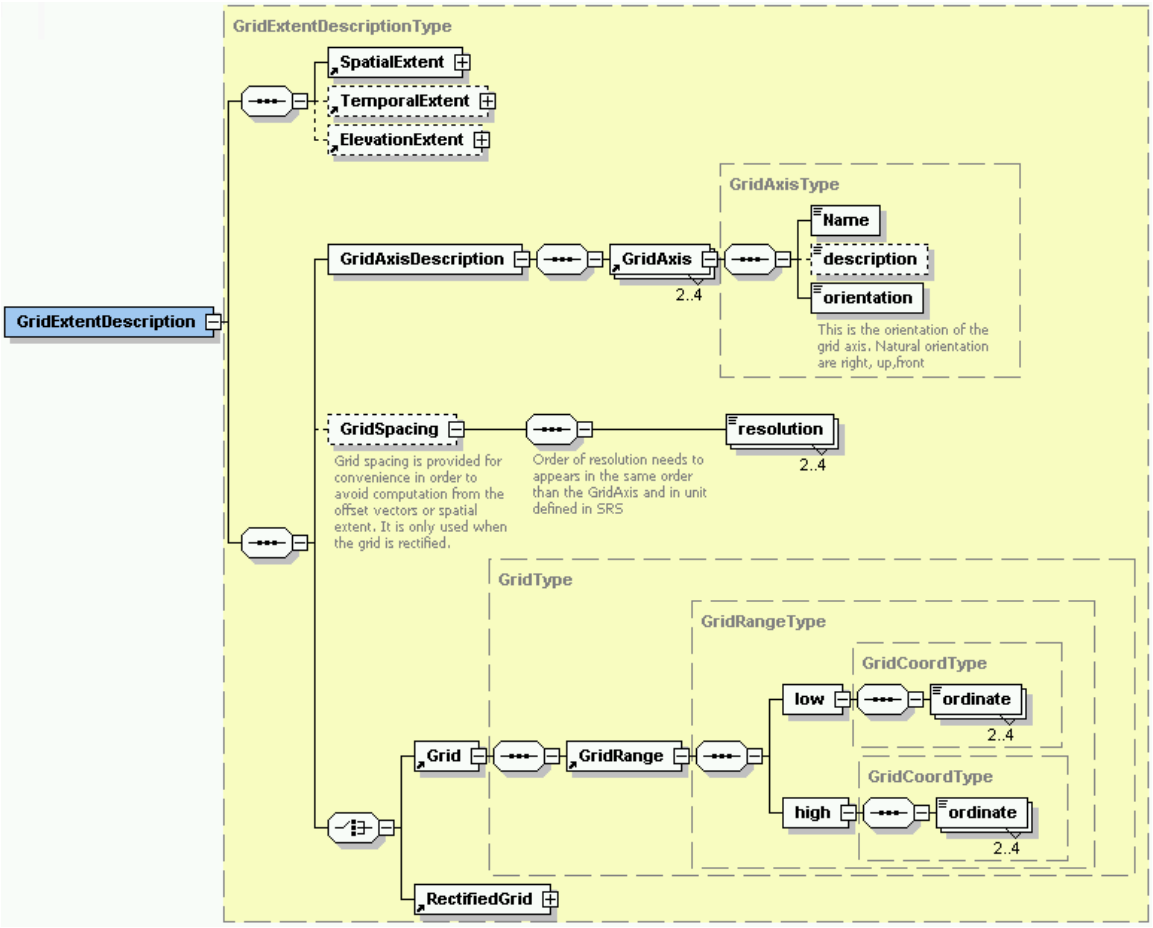

Figure 7 - Grid Coverage Layer domain description

The schema for **GridExtentDescription** adds three elements to the shared schema:

- **GridAxisDescription** is *required* It identifies each axis of the grid.
- **GridSpacing** is *required* for rectified grids, but *undefined* for other grids. It provides the ground resolution (pixel size or post spacing) along each dimension of the grid, expressed in the units of the (rectified) grid's Coordinate Reference System.
- *Either* **Grid** *or* **RectifiedGrid** is *required*. Both elements list the upper and lower bounds of the grid coordinates along each of the grid axes. (These are integer pixel or post coordinates, expressed in the grid's internal coordinate reference system. The lower bounds are commonly defined as zero.) **Grid** and **RectifiedGrid** both have two required attributes:
    - **dimension** (value: *2*, *3*, or *4*) indicates a 2-, 3-, or 4-dimensional grid.
    - **type** (value: *post* or *center*) indicates whether values are associated with the grid point locations (posts) or the areas between adjacent posts (centers).

RectifiedGrid describes grids, such as an orthophoto or Level 1G satellite image, whose grid coordinates in each dimension bear an affine relationship to those of a ground coordinate reference system. As shown in Figure 8, **RectifiedGrid** adds two required elements alongside the **GridRange**:

- **origin** lists the position of the grid's origin – that is, the (0,0,0,0) gridpoint – in ground coordinates.
- **offsets** lists the grid spacing along each axis in ground coordinates.

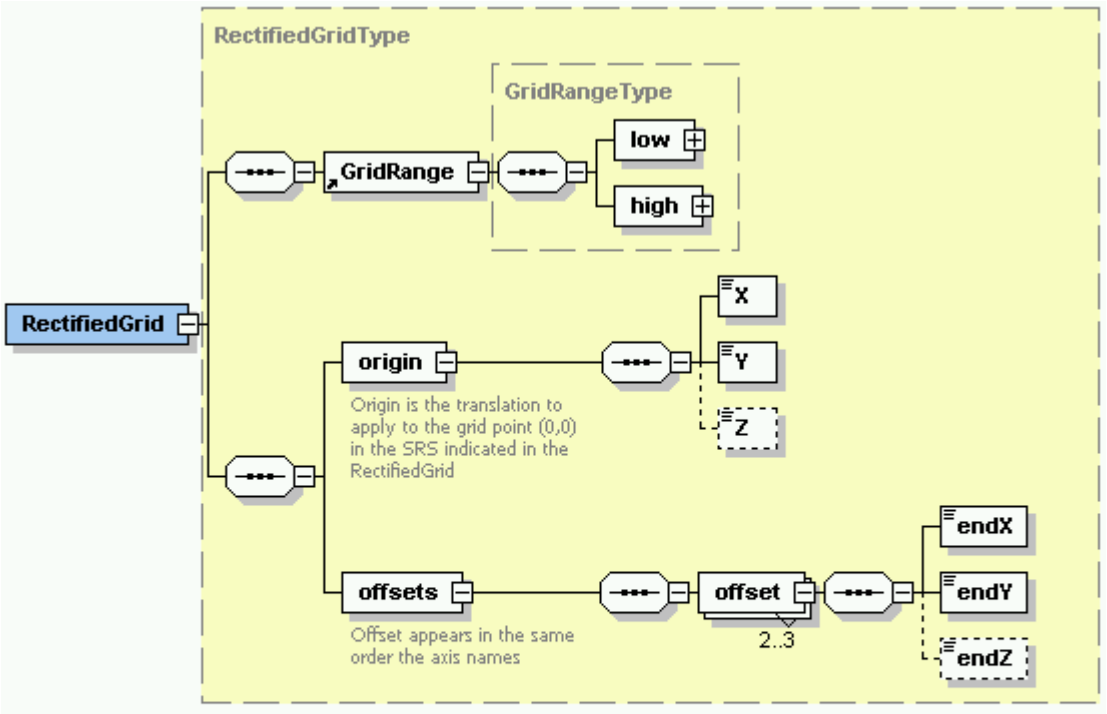

Figure 8 - Rectified grid description

RectifiedGrid also adds a third required attribute, **srsName**, to identify the ground coordinate system in which the grid's origin and offsets are expressed.

A third type of grid is also under consideration here (not yet finalized): the **RectifiableGrid** describes a grid whose axes are related to those of a ground coordinate system by a mathematical transformation, but not a simple affine relationship. In lieu of the origin and offsets of **RectifiedGrid**, the **RectifiableGrid** description has a **MathTransform** element, as shown in Figure 9.

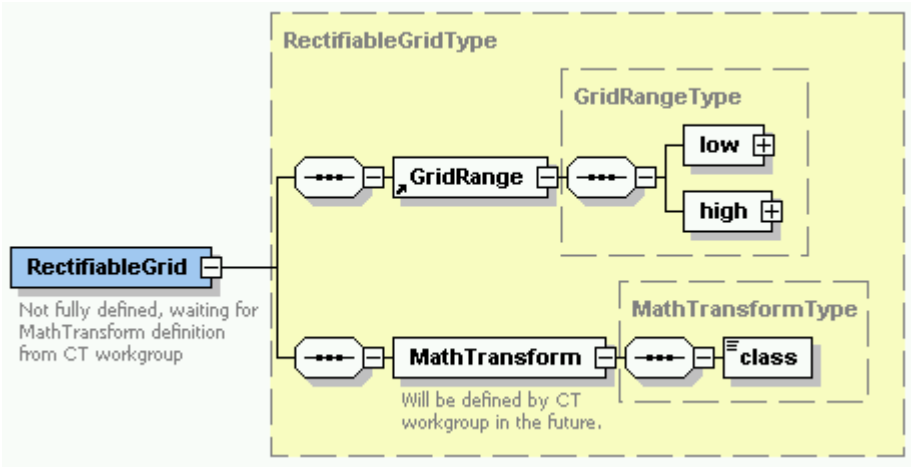

Figure 9 - Rectifiable grid description

MathTransform will list a transformation algorithm, based on the sensor model, geometry, and orientation. The syntax of this **MathTransform** element is not yet defined; it will draw on elements of the Sensor Markup Language IPR.

7.2.3.3 MultiPoint Coverage Layer

A second kind of coverage layer defines its domain as a series of points. These might be the locations of temperature readings, soil samples, or spot elevations. In addition to the shared elements (**SpatialExtent**, **TemporalExtent**, and **ElevationExtent**), this kind of coverage also lists two optional elements, as depicted in Figure 10:

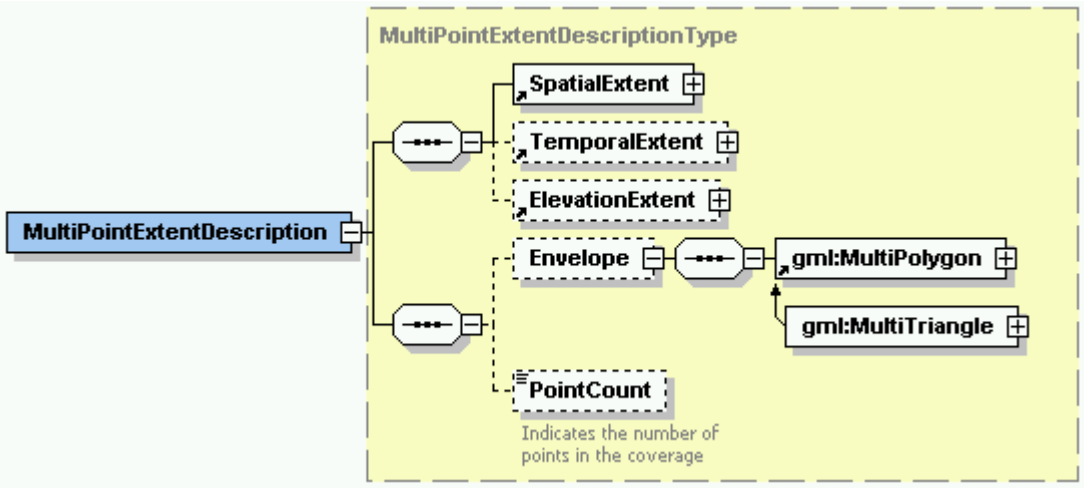

Figure 10 - Multi-Point Coverage Layer domain description

- **Envelope** lists one or more polygons that comprise the edge of available data points. This provides a more precise (tighter) description of the domain's spatial extent.

- **PointCount** lists the number of points in the domain of the coverage layer.

7.2.3.4 TIN Coverage Layer

A third type of coverage layer defines its domain as the triangles constructed from a series of points by Delaunay triangulation. (This kind of coverage often models a terrain by linear interpolation along these triangles: the ground surface is presumed to lie on the surface of the triangle that joins three nearby surveyed points.) Figure 11 shows the structure of such a domain description:

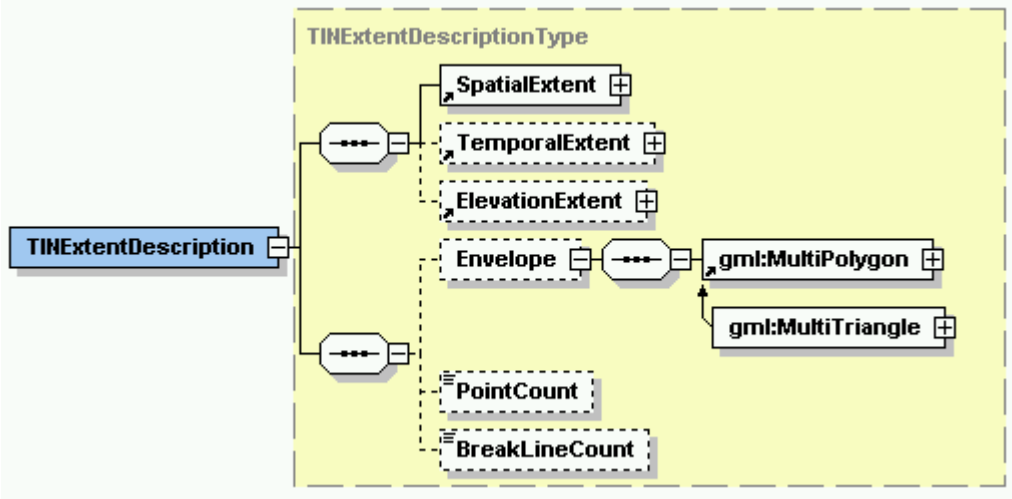

Figure 11 - TIN Coverage Layer domain description

The domain of a TIN coverage layer may be described very much like that of a multi-point coverage, with an **Envelope** and a **PointCount**. The TIN Coverage Layer provides one additional optional element, **BreakLineCount** , which lists the total number of breaklines defined in the TIN coverage layer. (Breaklines are sets of measured points where the gradient is known to change abruptly).

7.2.3.5 Segmented Curve Coverage Layer

Fourth, a segmented curve coverage layer defines its domain as a series of curves (that is, arcs, splines, or line-strings). This description might fit an inventory of streets and roads. As seen in Figure 12, this coverage layer is to curves what the multi-point coverage description is to points.

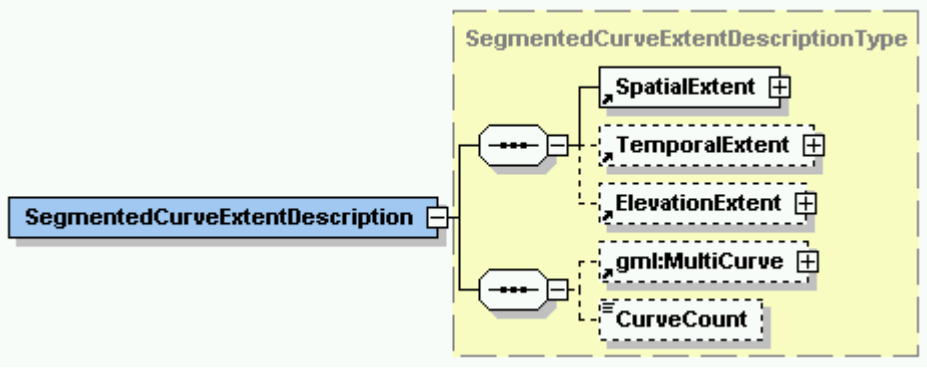

Figure 12 - Segmented Curve Coverage Layer domain description

7.2.3.6 Thiessen Polygon Coverage Layer

The last kind of coverage considered here defines its domain as polygons constructed from a series of known ("defining") points. Each polygon encloses not only one defining point, but all the positions in space that are closer to that defining point than to any other. The description of such a coverage is structured as in Figure 13.

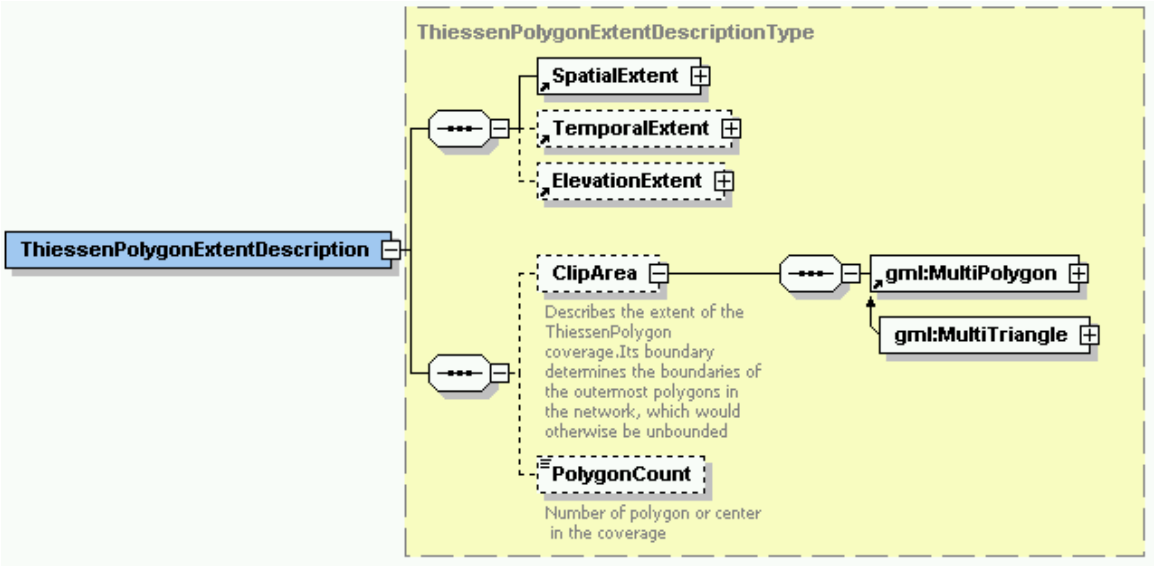

Figure 13 - Thiessen Polygon Coverage Layer domain description

The domain of a Thiessen polygon coverage usually includes a **ClipArea** element listing one or more polygons that comprise the outer edge of the available data. It also includes a polygonCount element that lists the total number of polygons in the coverage.

We probably need to talk about normal polygon coverages as well.

7.2.4 Range descriptions

The second element that all coverage types add to the generic **AbstractCoverageLayerType** is a **RangeSetDescription**. This element defines the properties (categories, measures, or values) assigned to each location in the domain. A location may have several unrelated properties attached to it (such as current land use, mean daily temperature, etc.): each of these is described in a separate Range Set. Any such property may be a scalar (numeric or text) value, such as population density, or a compound (vector or tensor) value, such as incomes by race, or radiances by wavelength.

7.2.4.1 Common Range Set model

In this version of WCS, the various coverage types all have an identically structured range set description (with the exception of GridCoverage, which adds a few image-specific elements, such as a histogram of values). This schema appears in Figure 14.

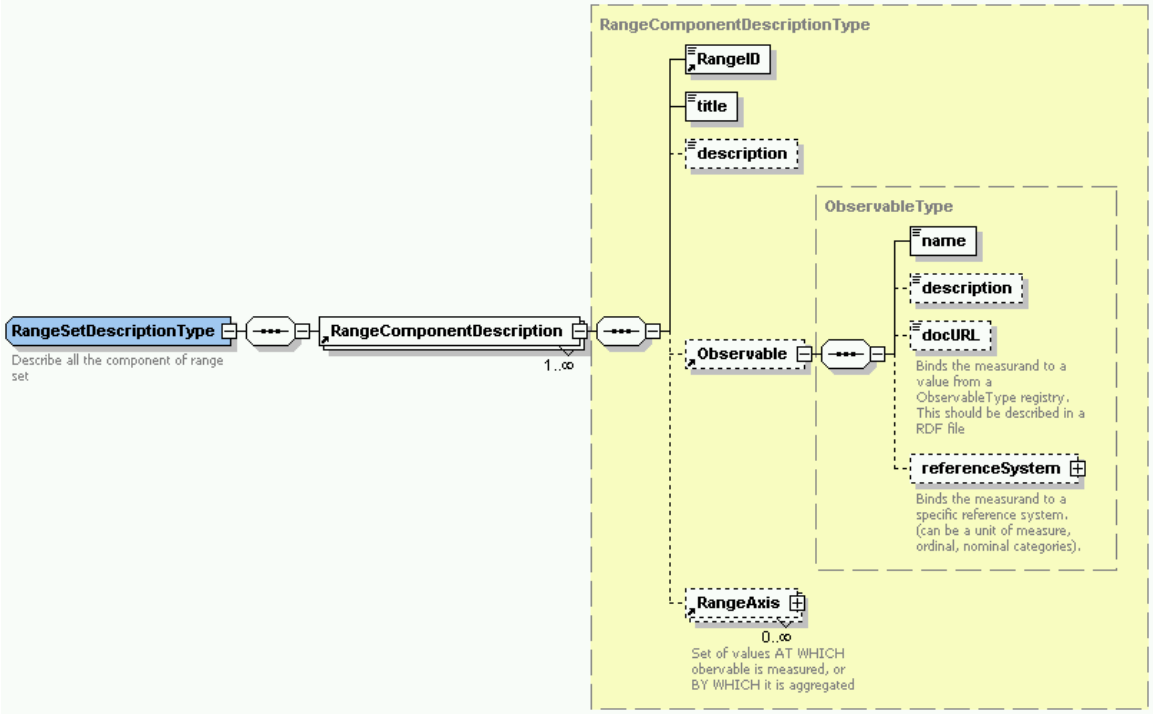

Figure 14 - Range Set Description

A single **RangeSetDescription** may have one or more **RangeComponentDescriptions** describing the different values associated with each location in the domain. Each **RangeComponentDescription** has the following elements:

- The *required* **RangeID** element uniquely identifies a range component within the coverage layer

- The *required* **title** element contains a human-readable string describing this range component, for presentation to the user in a menu.

- The *optional* **description** element contains a free-text description of the range component.

- The *optional* **Observable** element provides a structured description of the observations (quantities or properties) reported in this range component. This description consists of a **name**, a free-text **description**, a **docURL** (an index into a registry of observation types); and a **referenceSystem** that associates the reported values with real-world quantities or categories.

- The *optional* **RangeAxis** element is for compound observations. It describes and lists the values at which the range component reports properties, or the "bins" by which the range componet reports counts or other aggregate values.

Details on the last two of these elements follow.

7.2.4.1.1 Observable and referenceSystem

The **Observable** element sketched above articulates the information content of a range component in a coverage layer. (Complete details on Observations and Measurements may be found in the Observations and Measurements DIPR.) One of its key sub-elements, **referenceSystem**, describes how to interpret the values reported in the range set. The reference system may consist of a unit of measure, or a list of categories, as depicted in Figure 15.

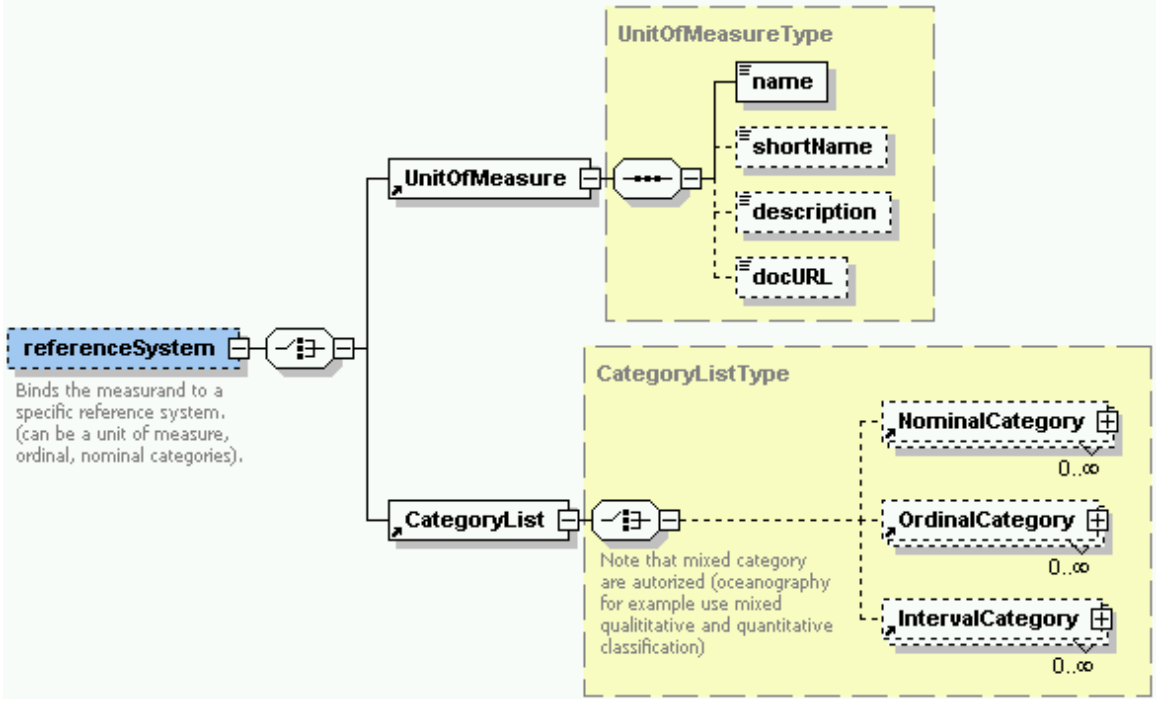

Figure 15 - Reference System

The **UnitOfMeasure** is used for measurements (and counts) of physical quantities, such as temperature in degrees Fahrenheit, or elevation in meters. It describes range values by means of a **name**, an abbreviation (**shortName**), a free-text **description**, and a **docURL** pointing into a registry of unit definitions.

A range component may also report properties as categories, rather than numeric measurements. The reference system for such properties is a list of categories identified by a unique **CategoryID**, a human-readable **title** (both *required*), a free-text **description**, and a **docURL** pointer into a registry, as depicted in Figure 16.

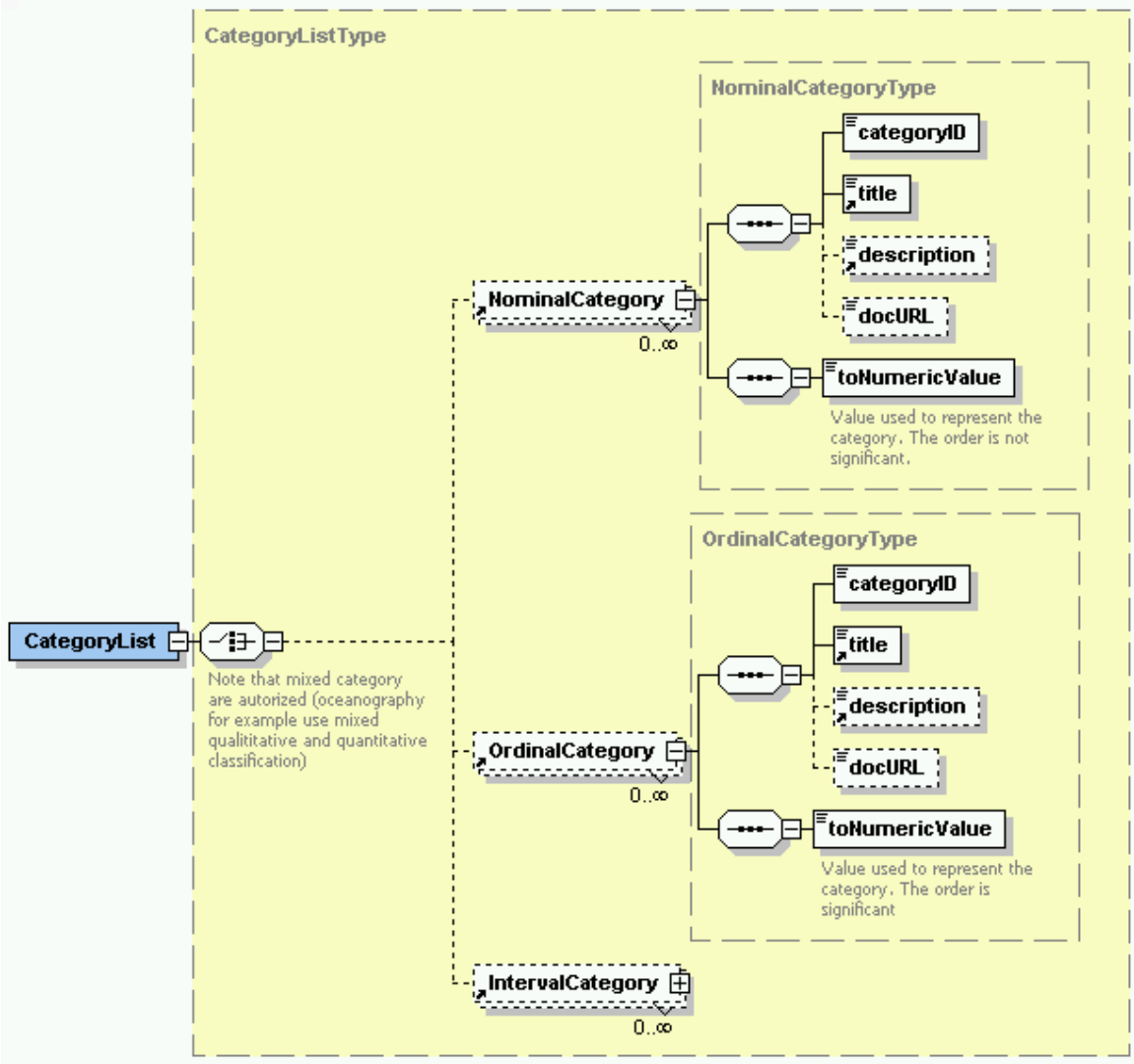

Figure 16 - Nominal and Ordinal Categories for Observations

Nominal and ordinal value descriptions report the numeric code used to represent this category (**toNumericValue**). Nominal and ordinal value descriptions differ only in whether the order of the categories is significant. It is significant for ordinal data (such as

school grades (excellent, good, …, failing) or vehicle size (economy, compact, …, fullsize); but not for nominal data (such as plant species or land ownership).

The third kind of category data, depicted in Figure 17, is interval data, for which the intervals correspond to brackets along some numeric dimension (e.g. age ranges 0-5, 6-10, 11-20, etc.). (Not quite sure what the **toValue** conversion is supposed to do.)

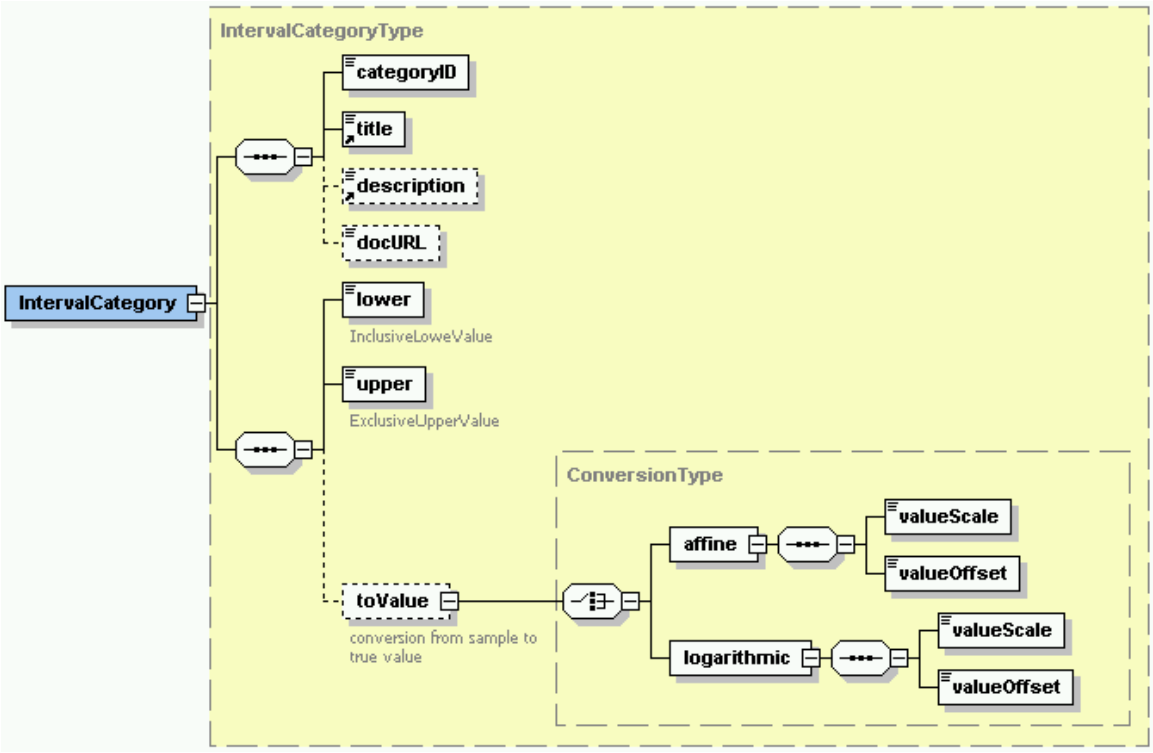

Figure 17 –Interval Categories for Observations

7.2.4.1.2 Compound observations and RangeAxis

A range component may report either simple scalar observations (terrain elevation, for instance, or yesterday's maximum temperature) or compound observations. Compound observations consist of a set of identically defined observations, each tied to a different value of a measurement parameter. Examples of compound observations include a spectral response (that is, brightness by wavelength in a multispectral image), age distribution (counts of people by age in a census table), or climate pattern (average rainfall by month in a climate database).

For a compound observation, in addition to describing the measurements themselves, it's often useful to describe the measurement parameter, and to list the parameter values for which (or "by which") measurements are available. This requires a **RangeAxis** element whose syntax appears in Figure 18.

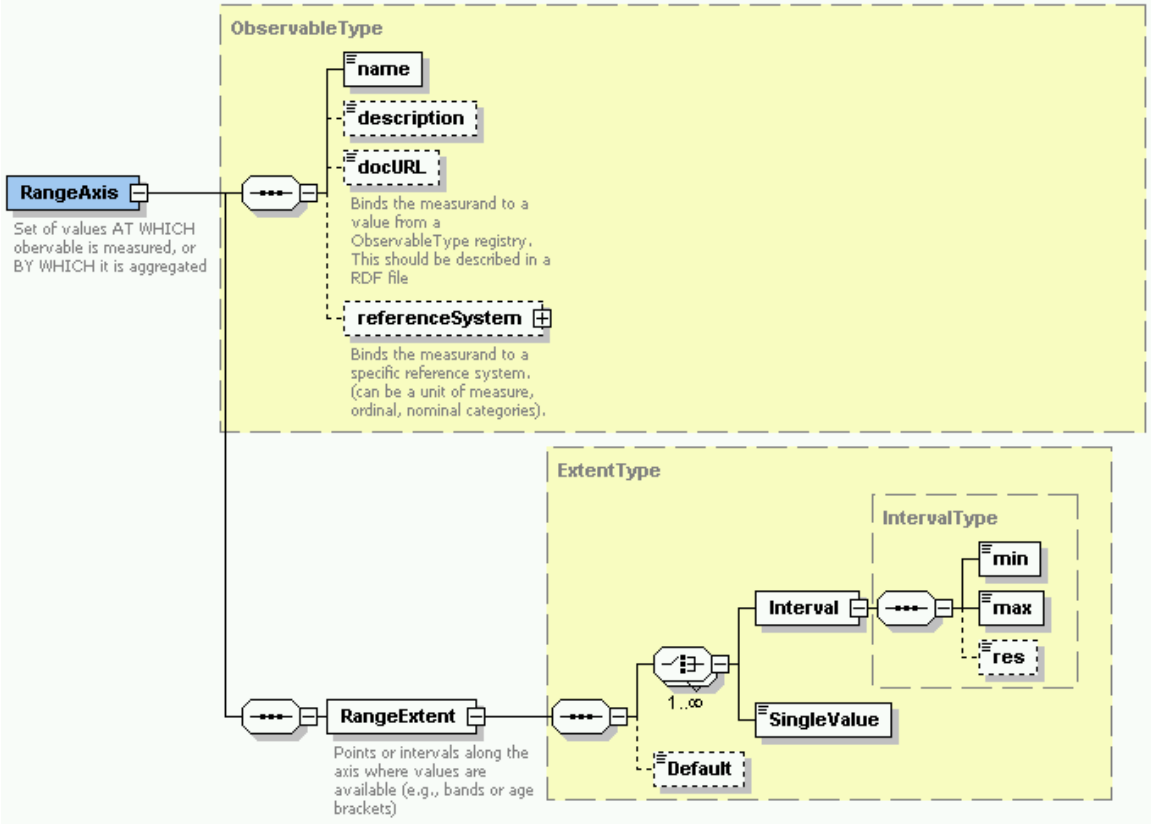

Figure 18 - Range Axis (for compound observables)

RangeAxis describes a measurement parameter using the same terms as those used to describe the measurements themselves (**name**, **description**, **docURL** and **referenceSystem)**. It also lists (in the **RangeExtent** element) the actual parameter values for which measurements are available.

For example, the **RangeAxis** for a multispectral image might indicate that it represents wavelengths in nanometers; and its **RangeExtent** would list the wavelengths of each image "band" for which radiance values are available.

A compound observable may have multiple **RangeAxis** elements, for observations related to parameter values along more than one axis (such as counts of wildlife tabulated both by size and by species).

7.2.4.1.3 Compound observable vs. multiple scalar observables

A compound observable is designed for observations that are identically defined – that report the same property, expressed in the same reference system. If a set of observables has any semantic variation, or any differences in the reference system, then the different observables belong in different range set components. For instance, a multispectral image in which some bands record emissive radiance, and others record reflective radiance, would require distinct range set components.

When several identically defined measurements are tied to an ordinal, interval, or ratio parameter, a compound measurement is often preferable to multiple scalar range set components. This is because the **RangeAxis** and **RangeExtent** elements of a compound measurement let clients retrieve subsets of the component observation by requesting intervals ("slices") along each range axis. For example, one may retrieve the near-infrared portions of a hyperspectral image by requesting that portion of the wavelength axis. Or, on may extract racial distribution among (only) the elderly from a table listing population counts by age and by race.

> In a future specification, compound values may also allow interpolation of measured observations along a range axis (where appropriate – e.g., for interval or ratio parameters, with values in narrow intervals separated by narrow gaps). For instance, hyperspectral imagery may allow interpolation of radiance values over wavelength if the spectral bands are narrow enough and close enough together. Similarly, population pyramids may allow interpolation of population over age if the age brackets are suitably defined.
>
> The range axis construct also anticipates "virtual coverages" (that is, real-time coverage servers that can adjust measurement parameter values on request) – such as an imaging sensor whose wavelength bands can be remote controlled, or a census data server that tabulates raw questionnaire data into age brackets of the user's choice.

When several identically defined measurements are tied to a nominal parameter (such as species) for which intervals ("slices") are undefined, a compound observable (such as counts by species) is functionally equivalent to multiple scalar range set components (count of species1, count of species2, etc.). In either case, range subset requests are limited to lists of individual values (white, hispanic, etc.). However, a compound observable does offer the notational convenience of describing the observable only once – a useful shorthand when the same observable is reported at many different values of a parameter.

7.2.4.2 Extensions defined for the range of a Grid Coverage Layer

In the case of a grid coverage layer, the range description adds several *optional* elements of the generic range component description, using a repeatable **GridRangeDescription**, depicted in Figure 19:

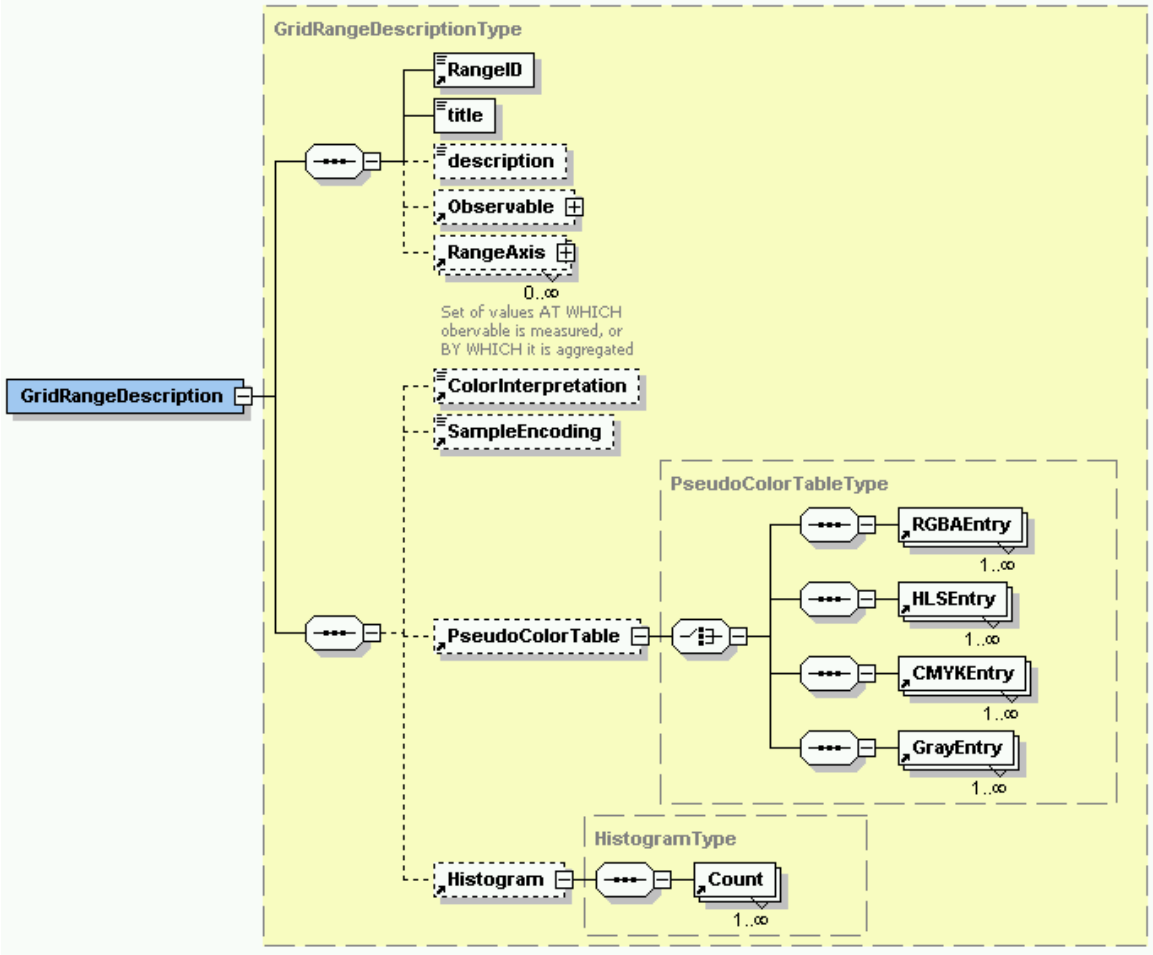

Figure 19 - Grid Range Description

- **ColorInterpretation** indicates how to read the image colors.
- **SampleEncoding** details the encoding of image samples
- **PseudoColorTable** lists indexed color values, expressed as red-green-blue-alpha values, or hue-lightness-saturation, cyan-magenta-yellow-black, or grayscale.
- **Histogram** records image statistics (mean and median value, etc.) and lists the counts and percentages of pixels in each of several brightness "bins."

7.3 Exceptions

In the event that the web coverage server encounters an error servicing a **GetCapabilities** request, it shall raise an exception as described in Section 6.4 above. The MIME type of the XML document containing the error message(s) **must** be application/vnd.ogc.se_xml.

8 GetCoverage operation (required)

The **GetCoverage** operation allows retrieval of coverages from a coverage layer. A WCS server processes a **GetCoverage** request and returns a result set to the client.

8.1 GetCoverage requests

A GetCoverage request may be encoded as key-value pairs, or as an XML document. The next two sections detail each of these encodings.

8.1.1 Key-Value Pair encoding

8.1.1.1 Overview

Table 1 describes the complete GetCoverage Request.

URL Component	Description
http://server_address/path/script?	URL of WCS server. *Required.*
VERSION=0.7	Request protocol version. *Required*.
REQUEST=GetCoverage	Request name: "GetCoverage". *Required*.
LAYER=LayerID	Name of an available coverage layer. *Required. Only one layer name allowed—despite the plural "layers".*
RANGE=Range1, Range2, …	Identifiers of one or more range set components defined on the requested layer. Not needed if the coverage layer has only one range set component.
SRS=srs_identifier	Spatial Reference System in which the request is expressed. *Required.*
RESPONSE_SRS= srs_identifier	Spatial Reference System in which to express responses. *Optional*; defaults to the request SRS.
BBOX=minx, miny, maxx, maxy [, minz, maxz]	Request a subset within the specified bounding box (lower left, upper right, optionally followed by z limits). These are expressed in the Spatial Reference System identified by SRS. *Required*.
ELEVATION=elev1,elev2,… *or* **ELEVATION=min/max/res, …**	Request a subset corresponding to the specified elevation points or intervals. *Optional* if a 2-D SRS is used and the queried coverage layer has a default elevation. *Not allowed* if 3-D CRS is used.
TIME=time1,time2,…	Request a subset corresponding to the specified

URL Component	Description
or **TIME=min/max/res, …**	time instants or intervals. *Optional* if SRS is used and a default time is defined for the queried layer.
***RANGE*:*PARAM*=min/max/res** *or* ***RANGE*:*PARAM*=val1,val2, …** or just ***PARAM* = …**	Request a subset by constraining parameter PARAM on a range component RANGE that has a compound observable. For instance: ***radiance:band=1,5,3*** *(radiance in bands 1, 5, 3)* ***count:age=0/18*** *(age distribution under 18 yrs.)* *Optional* if the chosen range component has default values for the parameter. The "RANGE:" prefix may be omitted if the coverage layer has only one range set component.
WIDTH = w (integer) **HEIGHT = h (integer)** **DEPTH =d (integer)**	Request a grid of the specified width (x), height (y), and depth (z) (integer number of gridpoints). *Either* these or RESX, RESY, RESZ are *required* for grid coverage layers.
RESX=*x (double)* **RESY=*y (double)*** **RESZ=*z (double)***	Subsample the spatial domain of a grid coverage layer to a specific resolution (expressed in the unit of the SRS). *Either* these or WIDTH, HEIGHT, DEPTH are *required* for grid coverage layers.
FORMAT=output_format	Output format of Coverage data file. *Required*.
EXCEPTIONS= application/vnd.ogc.se_xml	The format in which exceptions are to be reported by the Map Server. *Optional*.
Vendor-specific parameters	*Optional*.

Table 1 – The GetCoverage Request expressed as Key-Value Pairs.

8.1.1.2 VERSION=version

The Basic Service Elements section defines this parameter.

8.1.1.3 REQUEST=GetCoverage

The Basic Service Elements section defines this parameter. For GetCoverage, the value "GetCoverage" must be used.

8.1.1.4 LAYER=LayerID

The **LAYER** parameter requests a single coverage layer, identified in the Capabilities XML by a **LayerID** element.

Unlike WMS, a GetCoverage query must draw on only one coverage layer at a time. Combining coverages from different layers, visually or otherwise, is the client's job, not the server's. Future versions of WCS may address ways to combine coverages according to mathematical or logical operators (Boolean or other rule-based overlay).

8.1.1.5 RANGE=Range1,Range2, …

The **RANGE** parameter requests one or more of the range set components defined on the selected coverage layer in the Capabilities XML. Its value must match one or more **RangeID** elements under the selected coverage layer.

If the selected coverage layer has only one range set defined, this parameter is optional.

8.1.1.6 SRS

The SRS (Spatial Reference System) parameter is specified in the Basic Service Elements section.

GetCoverage requests must use this parameter to specify the coordinate reference system in which the query (BBOX) is expressed. The values of this request parameter **must** be one of those defined in a **SRS** or **QuerySRS/SRS** element under the requested coverage layer.

If the Capabilities XML's **QuerySRS** for a layer lists *only* ***"OGC:None"*** or ***"OGC:Swath"***, then the coverage layer must be queried in its internal pixel / local coordinate system. The Client **must** specify **SRS=OGC:None** or **SRS=OGC:Swath** (case-insensitive) in the GetCoverage request; the Server may issue a Service Exception otherwise.

8.1.1.7 RESPONSE_SRS

This parameter specifies the coordinate system in which the coverage response should be referenced. This parameter is optional; its value defaults to that of **SRS**. Thus, omitting it requests a coverage response referenced in the same coordinate reference system as the query (like WMS and WFS).

The value of this request parameter **must** be one of those defined in a **SRS** or **ResponseSRS/SRS** element defined under the requested coverage layer.

If the Capabilities XML's **SRS** or **ResponseSRS** for a coverage layer lists only ***"OGC:None"***, then the coverage layer does not have a well-defined spatial reference system and should not be shown in conjunction with other layers. The Client **must** specify **ResponseSRS=OGC:None** (case-insensitive) in the GetCoverage request; the Server may issue a Service Exception otherwise.

If the Capabilities XML's **SRS** or **ResponseSRS** for a coverage layer lists only ***"OGC:Swath"***, then the overage layer is only georeferenceable: its spatial reference system may be inferred from embedded tie-points or sensor metadata. The Client **may**

specify **ResponseSRS=OGC:Swath** (case-insensitive) in the GetCoverage request to request this non-georeferenced data.

8.1.1.8 BBOX

GetCoverage queries must express spatial constraints in at least 2-D (XY) coordinate space, using a **BBOX** parameter with comma-separated numbers (**minx, miny, maxx, maxy**), expressed in the spatial reference system given in the **SRS** parameter.

For any part of the coverage domain that is partly or entirely contained in the Bounding Box defined by BBOX, the server **should** return coverage data in the appropriate format.

In addition, if the Capabilities XML expresses the Layer's domain as a 3-D bounding box in a 3D coordinate reference system, then GetCoverage queries may use an extended **BBOX** syntax (**BBOX=minx,miny,maxx,maxy,*minz,maxz***), as described in the Basic Services Model. These extra numbers are optional: omitting *minz,maxz* requests the coverage layer's default elevation (if defined).

8.1.1.9 ELEVATION

If the Capabilities XML defines an **ElevationExtent** on the queried coverage layer, GetCoverage queries may use an **ELEVATION parameter** to constrain the request in elevation – thus supplementing a 2D XY BBOX.

The syntax for expressing Elevation constraints is specified in WMS 1.1 Annex C.

8.1.1.10 TIME

If the Capabilities XML defines a **TemporalExtent** on the queried coverage layer, GetCoverage queries may use a **TIME** parameter to constrain the request in time, thus supplementing a 2D XY or 3D XYZ BBOX.

The syntax for expressing Time constraints and date / time values is specified in WMS 1.1 Annexes B and C.

8.1.1.11 ***RANGE*:*PARAM*** **(or *PARAM*)**

If one of the selected range set components consists of a compound observable, GetCoverage queries may constrain the request along the parameter(s) of the compound observation. This is a variable parameter name, formed by concatenating one of the selected range set components (given in the **RANGE** parameter) with one of the RangeAxis names defined on that range set component, with a colon (':') in between.

For example, in Capabilities XML a given coverage layer might define a range set named "radiance" as a compound observable reported by wavelength intervals. A GetCoverage request might limit the request to visible wavelengths by specifying

RANGE=radiance,… *and*
RADIANCE:WAVELENGTH=650/700, 500/560, 430/500

(assuming “radiance” is a compound variable tied to a RangeAxis called “wavelength” that uses nanometers).

If the selected coverage layer has only one range set defined, the RANGE parameter is optional, as well as the corresponding parameter prefix. In the previous example, if “radiance” is in fact the only range set component defined on the selected coverage layer, then the constraint can be expressed simply as

WAVELENGTH=650/700, 500/560, 430/500

8.1.1.12 Grid size: WIDTH, HEIGHT, DEPTH

GetCoverage queries for gridded coverages may request coverage replies with a specific grid size. The parameters WIDTH, HEIGHT, DEPTH define the grid size (number of gridpoints or cells) along the three axes of the grid.

Either these parameters or RESX, RESY, RESZ are normally required for grid coverage layers. However, if the Capabilities XML reports *only* the Interpolation method “None” for the queried coverage layer, then GetCoverage queries must be for the full native resolution of the data: they may not use RESX, RESY, RESZ or WIDTH, HEIGHT, DEPTH to change the coverage resolution. In this case, BBOX alone is used for subsetting.

If the coverage layer has elevation extents expressed separately from its BoundingBox, then GetCoverage queries may use ELEVATION=… as described previously to request a particular extent and resolution.

8.1.1.13 Grid resolution: RESX, RESY, RESZ

GetCoverage queries for gridded coverages may request coverage replies in specific grid resolution. The parameters RESX and RESY define the grid-cell size along the first and second axes of the coordinate reference system given in SRS or RESPONSE_SRS.

If the RESPONSE_SRS is a 3D spatial reference system, then the additional RESZ parameter may be used to specify the desired resolution along the third axis of that spatial reference system.

Either these parameters or WIDTH, HEIGHT, DEPTH are normally required for grid coverage layers. However, if the Capabilities XML reports *only* the Interpolation method “None” for the queried coverage layer, then GetCoverage queries must be for the full native resolution of the data: they may not use RESX, RESY, RESZ or WIDTH, HEIGHT, DEPTH to change the coverage resolution. In this case, BBOX alone is used for subsetting.

If the coveage layer has elevation extents expressed separately from its BoundingBox, then GetCoverage queries may use ELEVATION=… as described previously to request a particular extent and resolution.

8.1.1.14 FORMAT

The value of this parameter must be one of those listed in the **Format** element of the desired coverage layer in Capabilities XML. The entire MIME type string in the **MIMEType** sub-element is used as the value of the FORMAT parameter. In an HTTP environment, the MIME type **must** be set on the returned object using the "Content-type" entity header.

8.1.1.15 EXCEPTIONS

A Web Coverage Service **must** offer the exception reporting format "**application/vnd.ogc.se_xml"** by listing it in a <Exceptions><Format> element in its Capabilities XML. The entire MIME type string in <Format> is used as the value of the EXCEPTIONS parameter.

Errors are reported using Service Exception XML, as specified in WMS 1.1 Section 6.7 and Annex A.3. This is the default exception format if none is specified in the request.

8.1.2 XML encoding

Figure 20 depicts a schema for the XML encoding of GetCoverage.

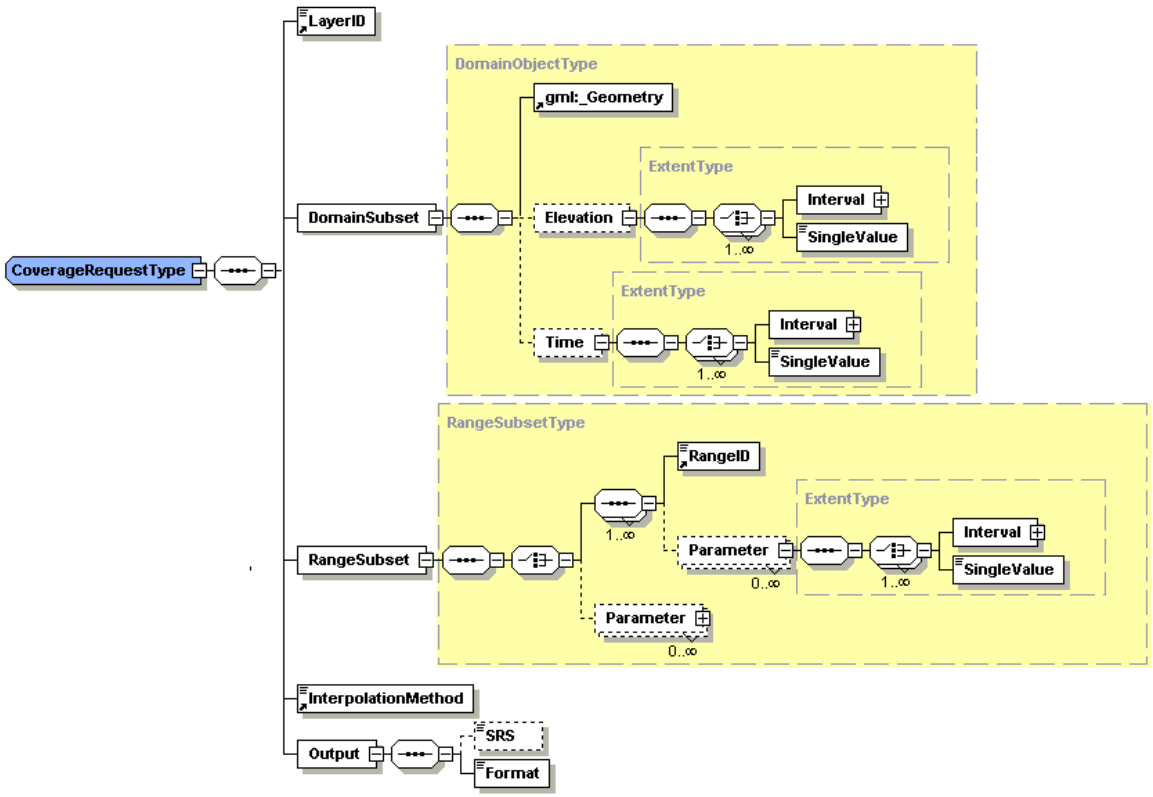

Figure 20 - GetCoverage Request Type (general)

CoverageRequestType requires a **version** attribute and 5 sub-elements:

- **LayerID** specifies one of the coverage layers listed in the Capabilities XML.
- **DomainSubset** expresses what subset of the spatial and temporal to retrieve. In general, the spatial subset may be any geometry; indeed, **DomainSubset** has a GML **_Geometry** abstract element. **DomainSubset** also allows subsets on **Elevation** and **Time** (values or intervals).
- **RangeSubset** expresses what subset of the range to retrieve. If the coverage layer has multiple range set components listed in Capabilities XML, this section requests one (or several) of those, identified by **RangeID**. If the selected coverage layer has only one range component (whether scalar or compound), then the **RangeID** element may be omitted.

  Clients may fetch the range component in its entirety; or in the case of a compound measurement, they may request subsets by **Parameter** values. The **Parameter name** attribute must match that of a **RangeAxis** defined on the given range component in the Capabilities XML. The requested **Parameter** value(s) must be among those listed in the **RangeExtent** for the requested **RangeAxis** in the Capabilities XML.
- **InterpolationMethod** specifies what type of interpolation to use for resampling coverage values over the spatial domain. This must be one of those listed for this

coverage layer in Capabilities XML. Options are *"linear"*, *"bilinear"*, *"bicubic"*, *"lost area"*, *"barycentric"*, *"piecewise constant"*, and *"none"*.

- **Output** asks for coverage responses to be expressed a particular Spatial Reference System (**SRS**) and encoded in a particular **Format**.

Figure 21 shows a specialization of **GetCoverage** for requests against a grid coverage layer.

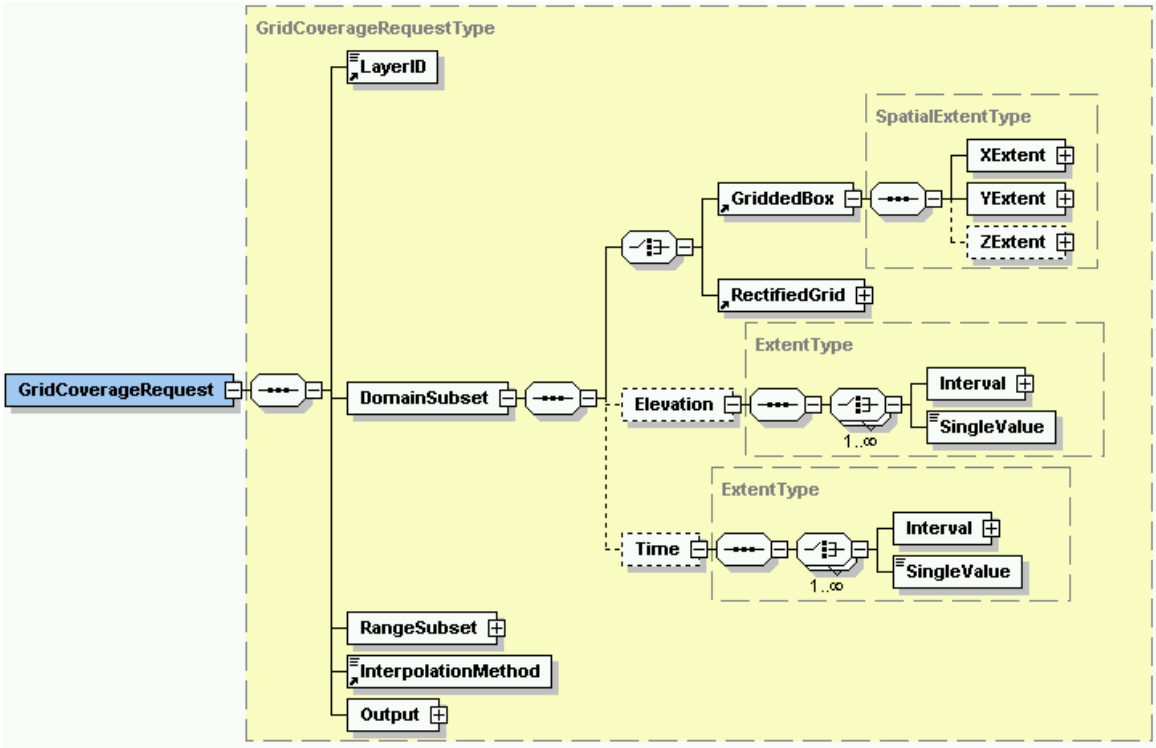

Figure 21 – Grid Coverage Request

This schema conforms to the generic **CoverageRequest** type, except that it replaces the abstract **gml:_Geometry** with either a 2-D or 3-D **GriddedBox**, or a **RectifiedGrid**.

Future work may define additional specialized request schemas.

8.2 GetCoverage response

The response to a valid GetCoverage request **must** be a coverage extracted from the coverage layer requested, with the specified spatial reference system, bounding box, size, and format.

An invalid GetCoverage request **must** yield an error output in the requested Exceptions format (or a network protocol error response in extreme cases).

In an HTTP environment, the MIME type of the returned value's Content-type entity header **must** match the format of the return value.

8.2.1 Coverage encoding

WCS does not prescribe any particular encoding for coverage replies. Here are a few widely-used encodings:

- GeoTIFF
- HDF-EOS
- DTED
- NITF

The above list is not normative; but merely a “starter set” of examples; an individual server may extend this list within its own Capabilities XML response.

8.2.2 Exceptions

For WCS, the **Exceptions** tag in the Capabilities response (and its counterpart EXCEPTIONS parameter in a GetCoverage query) is optional; if present, it must have the value " **application/vnd.ogc.se_xml** ".

A Web Coverage Server throwing an exception shall adhere to the value of the EXCEPTIONS parameter. Nonetheless, a Web Coverage server may, due to circumstances beyond its control, return nothing (this might result from the HTTP server’s behavior caused by a malformed request, by an invalid HTTP request, by access violations, or any of several other conditions). Web Coverage clients should be prepared for this eventuality.

8.2.3 Approximate Responses to GetCoverage

Finally, a WCS Server may in some cases return only an *approximation* to the client's request. For instance, it may not be able to provide the exact resolution or extent requested along various spatio-temporal or sample dimensions. However, any coverage encoding will clearly specify its actual resolution or spatial extent. Clients, therefore, should not make *a priori* assumptions about the resolution or extent of the returned coverage; instead, clients should *(i)* consult the coverage layer description in Capabilities XML before requesting coverages from it, and *(ii)* inspect each coverage reply before using or rendering it.

9 DescribeCoverageLayer operation (optional)

The **DescribeCoverageLayer** operation lets clients request a full description of any coverage layer served by a particular WCS server. The server responds with an XML document describing one or more coverage layers served by the WCS.

In the current version of the WCS interface, the Capabilities XML file contains detailed descriptions of all available coverage layers, along with detailed service descriptions. **DescribeCoverageLayer** is a convenience operation that fetches one or more child elements of **CoverageLayerList** from the Capabilities XML file.

The **GetCapabilities** operation could retrieve the same information; however, the ability to inquire about only one or a few coverage layers is useful for WCS servers with many coverage layers – and for integration with catalogs that store only a layer name and brief summary description.

Furthermore, future versions of WCS may simplify the descriptions in the Capabilities XML file, and provide more complete descriptive information only via **DescribeCoverageLayer**.

9.1 DescribeCoverageLayer requests

A **DescribeCoverageLayer** request lists the coverage layers to be described, identified by the content of their **LayerID** element. A request that lists no coverage layers shall be interpreted as requesting descriptions of all coverage layers that a WCS can serve.

9.1.1 Key-Value Pair encoding

Table 5.1 describes the complete **DescribeCoverageLayer** request in its HTTP GET form.

URL Component	Description
http://server_address/path/script?	URL of WCS server. *Required.*
VERSION=0.7	Request protocol version. *Required*.
REQUEST=DescribeCoverageLayer	Request name: **DescribeCoverageLayer.** *Required*.
LAYER=layer1,layer2, …	A comma-separated list of coverage layer to describe (identified by their **LayerID**s). *Optional.* Default is all coverage layers.

9.1.2 XML encoding

The DescribeCoverageLayer Schema is a simple one:

```
<xs:element name="DescribeCoverageLayer">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="LayerID" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="version" type="xs:string" use="optional" fixed="0.7"/>
    </xs:complexType>
</xs:element>
```

The top-level element, **DescribeCoverageLayer**, has one attribute, **version**, denoting the version number of the protocol. **DescribeCoverageLayer** has one optional sub-element, **LayerID**, denoting which coverage layer(s) to describe. If the **LayerID** element is absent, the server must return the entire **CoverageLayerList** from the Capabilities XML document.

A simple XML example follows – this is for requesting description of three different coverage layers on a single WCS server.

```
<DescribeCoverageLayer version="0.7">
   <LayerID>Landsat_TM_Mosaic</LayerID>
   <LayerID>WMO_Daily_Temps</LayerID>
   <LayerID>Census_population_tables</LayerID>
</DescribeCoverageLayer>
```

9.2 DescribeCoverageLayer response

In response to a **DescribeCoverageLayer** request, a WCS will return an XML document whose top-level element is a **CoverageLayerList** containing descriptions for all (and only) the requested coverage layers.

10 References

Annex A
(normative)

XML Schemas

A.1 WCS Capabilities Schema

The full listing of the WCS schema follows. It imports the GML schema from http://gws.pcigeomatics.com/OWS/WCS/geometry.xsd.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com)
     by Stephane Fellah (PCI Geomatics) and John D Evans (Global Science & Technology, Inc) -->
<xs:schema targetNamespace="http://www.opengis.net/wcs"
     xmlns="http://www.opengis.net/wcs"
     xmlns:xs="http://www.w3.org/2001/XMLSchema"
     xmlns:xlinks="http://www.opengis.net/gml"
     xmlns:gml="http://www.opengis.net/gml"
     elementFormDefault="qualified" version="0.2">
     <xs:annotation>
          <xs:appinfo>wcs.xsd v0.7 2002-04-05</xs:appinfo>
          <xs:documentation xml:lang="en"> This schema defines the common type and elements used by the OGC
web coverage server.</xs:documentation>
     </xs:annotation>
     <!-- ============================================================
                              includes and imports
     ============================================================ -->
     <xs:import namespace="http://www.opengis.net/gml"
                    schemaLocation="http://gws.pcigeomatics.com/OWS/WCS/geometry.xsd"/>
     <!-- ============================================================
                    Definition of coverage layer complex types
============================================================ -->
     <!--CoverageLayerList definition-->
     <xs:element name="CoverageLayerList" type="CoverageLayerListType"/>
     <xs:element name="title" type="xs:string"/>
     <xs:element name="description" type="xs:string"/>
     <xs:complexType name="CoverageLayerListType">
          <xs:annotation>
               <xs:documentation>Describe the list of coverage layers served by the WCS. Use for the
capabilities</xs:documentation>
          </xs:annotation>
          <xs:choice maxOccurs="unbounded">
               <xs:element ref="GridCoverageLayer" minOccurs="0"/>
               <xs:element ref="TINCoverageLayer" minOccurs="0"/>
               <xs:element ref="MultiPointCoverageLayer" minOccurs="0"/>
               <xs:element ref="SegmentedCurveCoverageLayer" minOccurs="0"/>
               <xs:element ref="ThiessenPolygonCoverageLayer" minOccurs="0"/>
          </xs:choice>
     </xs:complexType>
     <xs:complexType name="AbstractCoverageLayerType" abstract="true">
          <xs:annotation>
               <xs:documentation>This  abstract class is the base class used to describe coverages provided by a
WCS.</xs:documentation>
          </xs:annotation>
          <xs:complexContent>
               <xs:extension base="LayerType">
                    <xs:sequence minOccurs="0">
```

```
                    <xs:element ref="SupportedInterpolationList">
                        <xs:annotation>
                            <xs:documentation>In case of discrete coverage, NONE must be
used.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="GridCoverageLayerType">
        <xs:annotation>
            <xs:documentation>This type describe a layer serving a grid coverage </xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="AbstractCoverageLayerType">
                <xs:sequence>
                    <xs:element ref="GridExtentDescription"/>
                    <xs:element name="RangeSetDescription">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="GridRangeDescription" maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="TINCoverageLayerType">
        <xs:annotation>
            <xs:documentation>This type describes a layer serving a TIN coverage </xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="AbstractCoverageLayerType">
                <xs:sequence>
                    <xs:element ref="TINExtentDescription"/>
                    <xs:element ref="RangeSetDescription"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="MultipointCoverageLayerType">
        <xs:annotation>
            <xs:documentation>This type describes a layer serving a multipoint  coverage </xs:documentation>
            <xs:documentation>To do</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="AbstractCoverageLayerType">
                <xs:sequence>
                    <xs:element ref="MultiPointExtentDescription"/>
                    <xs:element ref="RangeSetDescription"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="SegmentedCurveCoverageLayerType">
        <xs:annotation>
            <xs:documentation>This type describe a layer serving a segmented curve coverage
</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="AbstractCoverageLayerType">
                <xs:sequence>
                    <xs:element ref="SegmentedCurveExtentDescription"/>
```

```
                    <xs:element ref="RangeSetDescription"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="ThiessenCoverageLayerType">
        <xs:annotation>
            <xs:documentation>This type describe a layer serving a thiessen coverage </xs:documentation>
            <xs:documentation>To do</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="AbstractCoverageLayerType">
                <xs:sequence>
                    <xs:element ref="ThiessenPolygonExtentDescription"/>
                    <xs:element ref="RangeSetDescription"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <!--Coverage Layer Global elements-->
    <xs:element name="GridCoverageLayer" type="GridCoverageLayerType">
        <xs:annotation>
            <xs:documentation>Grid coverage layer element</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="TINCoverageLayer" type="TINCoverageLayerType"/>
    <xs:element name="ThiessenPolygonCoverageLayer" type="ThiessenCoverageLayerType"/>
    <xs:element name="MultiPointCoverageLayer" type="MultipointCoverageLayerType"/>
    <xs:element name="SegmentedCurveCoverageLayer" type="SegmentedCurveCoverageLayerType"/>
    <!--Complex type for WCS requests -->
    <xs:complexType name="CoverageRequestType">
        <xs:sequence>
            <xs:element ref="LayerID"/>
            <xs:element name="DomainSubset" type="DomainObjectType"/>
            <xs:element name="RangeSubset" type="RangeSubsetType"/>
            <xs:element ref="InterpolationMethod"/>
            <xs:element name="Output">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="SRS" type="xs:string" minOccurs="0"/>
                        <xs:element name="Format" type="xs:string"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="version" type="xs:string" fixed="1.0.0"/>
    </xs:complexType>
    <xs:complexType name="GridCoverageRequestType">
        <xs:complexContent>
            <xs:restriction base="CoverageRequestType">
                <xs:sequence>
                    <xs:element ref="LayerID"/>
                    <xs:element name="DomainSubset">
                        <xs:complexType>
                            <xs:complexContent>
                                <xs:restriction base="DomainObjectType">
                                    <xs:sequence>
                                        <xs:choice>
                                            <xs:element ref="GriddedBox">
                                                <xs:annotation>
                                                    <xs:documentation>Used for georeferenced
coverage</xs:documentation>
                                                </xs:annotation>
                                            </xs:element>
                                            <xs:element ref="RectifiedGrid"/>
```

```
                                    </xs:choice>
                                    <xs:element name="Elevation" type="ExtentType" minOccurs="0"/>
                                    <xs:element name="Time" type="ExtentType" minOccurs="0"/>
                                </xs:sequence>
                            </xs:restriction>
                        </xs:complexContent>
                    </xs:complexType>
                </xs:element>
                <xs:element name="RangeSubset">
                    <xs:complexType>
                        <xs:complexContent>
                            <xs:extension base="RangeSubsetType"/>
                        </xs:complexContent>
                    </xs:complexType>
                </xs:element>
                <xs:element ref="InterpolationMethod"/>
                <xs:element name="Output">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="SRS" type="xs:string" minOccurs="0"/>
                            <xs:element name="Format" type="xs:string"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:restriction>
    </xs:complexContent>
</xs:complexType>
<!--Global elements for WCS requests-->
<xs:element name="CoverageRequest" type="CoverageRequestType" abstract="true"/>
<xs:element name="GridCoverageRequest" type="GridCoverageRequestType"/>
<!--Common types and elements-->
<!--DomainSetExtentDescription elements-->
<xs:element name="DomainSetExtentDescription" type="DomainSetExtentDescriptionType"/>
<xs:element name="GridExtentDescription" type="GridExtentDescriptionType"
substitutionGroup="DomainSetExtentDescription"/>
<xs:element name="MultiPointExtentDescription" type="MultiPointExtentDescriptionType"
substitutionGroup="DomainSetExtentDescription"/>
<xs:element name="TINExtentDescription" type="TINExtentDescriptionType"
substitutionGroup="DomainSetExtentDescription"/>
<xs:element name="SegmentedCurveExtentDescription" type="SegmentedCurveExtentDescriptionType"
substitutionGroup="DomainSetExtentDescription"/>
<xs:element name="ThiessenPolygonExtentDescription" type="ThiessenPolygonExtentDescriptionType"
substitutionGroup="DomainSetExtentDescription"/>
<!--Complex types for DomainSetExtentDescription-->
<xs:complexType name="DomainSetExtentDescriptionType">
    <xs:annotation>
        <xs:documentation>Description of the native domain set of the coverage</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element ref="SpatialExtent"/>
        <xs:element ref="TemporalExtent" minOccurs="0">
            <xs:annotation>
                <xs:documentation>Use for temporal slices. The slices needs to be ordered from the lowest to
the highest </xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element ref="ElevationExtent" minOccurs="0">
            <xs:annotation>
                <xs:documentation>Use for elevation slices</xs:documentation>
            </xs:annotation>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="temporal" type="xs:boolean" use="optional"/>
    <xs:attribute name="dimension" type="xs:positiveInteger" use="optional"/>
```

```
    </xs:complexType>
    <xs:complexType name="GridExtentDescriptionType">
        <xs:annotation>
            <xs:documentation>Describe the domain of a grid coverage</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="DomainSetExtentDescriptionType">
                <xs:sequence>
                    <xs:element name="GridAxisDescription">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="GridAxis" minOccurs="2" maxOccurs="4"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="GridSpacing" minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>Grid spacing is provided for convenience in order to avoid
computation from the offset vectors or spatial extent. It is only used when the grid is rectified.</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:sequence>
                                <xs:annotation>
                                    <xs:documentation>Order of resolution needs to appears in the same order
than the GridAxis and in unit defined in SRS</xs:documentation>
                                </xs:annotation>
                                <xs:element name="resolution" type="xs:double" minOccurs="2" maxOccurs="4"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:choice>
                        <xs:element ref="Grid"/>
                        <xs:element ref="RectifiedGrid"/>
                    </xs:choice>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="MultiPointExtentDescriptionType">
        <xs:annotation>
            <xs:documentation>Describe the domain of a multipoint coverage</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="DomainSetExtentDescriptionType">
                <xs:sequence>
                    <xs:element name="Envelope" minOccurs="0">
                        <xs:complexType>
                            <xs:complexContent>
                                <xs:restriction base="gml:GeometryAssociationType">
                                    <xs:sequence>
                                        <xs:element ref="gml:MultiPolygon"/>
                                    </xs:sequence>
                                </xs:restriction>
                            </xs:complexContent>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="PointCount" type="xs:positiveInteger" minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>Indicates the number of points in the
coverage</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
```

```
    </xs:complexType>
    <xs:complexType name="TINExtentDescriptionType">
        <xs:annotation>
            <xs:documentation>Describe the domain of a TIN coverage</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="DomainSetExtentDescriptionType">
                <xs:sequence>
                    <xs:element name="Envelope" minOccurs="0">
                        <xs:complexType>
                            <xs:complexContent>
                                <xs:restriction base="gml:GeometryAssociationType">
                                    <xs:sequence>
                                        <xs:element ref="gml:MultiPolygon"/>
                                    </xs:sequence>
                                </xs:restriction>
                            </xs:complexContent>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="PointCount" type="xs:positiveInteger" minOccurs="0"/>
                    <xs:element name="BreakLineCount" type="xs:positiveInteger" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="SegmentedCurveExtentDescriptionType">
        <xs:annotation>
            <xs:documentation>Describe the domain of segemented curve coverage</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="DomainSetExtentDescriptionType">
                <xs:sequence>
                    <xs:element ref="gml:MultiCurve" minOccurs="0"/>
                    <xs:element name="CurveCount" type="xs:positiveInteger" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="ThiessenPolygonExtentDescriptionType">
        <xs:annotation>
            <xs:documentation>Describe the domain of a thiessen polygon coverage</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="DomainSetExtentDescriptionType">
                <xs:sequence>
                    <xs:element name="ClipArea" minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>Describes the extent of the ThiessenPolygon coverage.Its
boundary determines the boundaries of the outermost polygons in the network, which would otherwise be
unbounded</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:complexContent>
                                <xs:restriction base="gml:GeometryAssociationType">
                                    <xs:sequence>
                                        <xs:element ref="gml:MultiPolygon"/>
                                    </xs:sequence>
                                </xs:restriction>
                            </xs:complexContent>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="PolygonCount" type="xs:positiveInteger" minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>Number of polygon or center  in the coverage</xs:documentation>
                        </xs:annotation>
```

```
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <!--Common types for describing extent-->
    <xs:element name="_DomainExtent" type="DomainExtentType" abstract="true"/>
    <xs:element name="SpatialExtent" type="SpatialExtentType" substitutionGroup="_DomainExtent"/>
    <xs:element name="TemporalExtent" substitutionGroup="_DomainExtent">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="ExtentType">
                    <xs:sequence minOccurs="0">
                        <xs:element name="Default" type="xs:string"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="ElevationExtent" substitutionGroup="_DomainExtent">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="ExtentType">
                    <xs:sequence minOccurs="0">
                        <xs:element name="Default"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:complexType name="DomainExtentType" abstract="true">
        <xs:annotation>
            <xs:documentation>Asbtract type to define a domain extent</xs:documentation>
        </xs:annotation>
    </xs:complexType>
    <xs:complexType name="SpatialExtentType">
        <xs:annotation>
            <xs:documentation>Describes the spatial extent of a coverage</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="DomainExtentType">
                <xs:sequence>
                    <xs:element name="XExtent" type="IntervalType"/>
                    <xs:element name="YExtent" type="IntervalType"/>
                    <xs:element name="ZExtent" type="IntervalType" minOccurs="0"/>
                </xs:sequence>
                <xs:attribute name="srsName" type="xs:anyURI" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="ElevationExtentType">
        <xs:annotation>
            <xs:documentation>Describes explicitly the elevation extent of the coverage</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="DomainExtentType">
                <xs:choice>
                    <xs:element name="ElevationInterval" type="ElevationIntervalType" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:element name="ElevationSeries" type="ElevationSeriesType" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:element name="defaultElevation" type="xs:double" minOccurs="0"/>
                </xs:choice>
            </xs:extension>
        </xs:complexContent>
```

```
    </xs:complexType>
    <xs:complexType name="ExtentType">
        <xs:annotation>
            <xs:documentation>Describes explicitly the elevation extent of the coverage</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:choice maxOccurs="unbounded">
                <xs:element name="Interval" type="IntervalType"/>
                <xs:element name="SingleValue" type="xs:string"/>
            </xs:choice>
        </xs:sequence>
        <xs:attribute name="uom" type="xs:string" use="optional"/>
    </xs:complexType>
    <xs:complexType name="IntervalType">
        <xs:annotation>
            <xs:documentation>Define an interval along an axis</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="min" type="xs:string"/>
            <xs:element name="max" type="xs:string"/>
            <xs:element name="res" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ElevationIntervalType">
        <xs:annotation>
            <xs:documentation>Describe a serie of elevations at regular interval</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="minZ" type="xs:double"/>
            <xs:element name="maxZ" type="xs:double"/>
            <xs:element name="resZ" type="xs:double"/>
        </xs:sequence>
        <xs:attribute name="uom" type="xs:string"/>
    </xs:complexType>
    <xs:complexType name="ElevationSeriesType">
        <xs:annotation>
            <xs:documentation>Describes a series of elevations</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="Elevation" type="xs:double" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="uom" type="xs:string"/>
    </xs:complexType>
    <xs:complexType name="TemporalExtentType">
        <xs:annotation>
            <xs:documentation>Describe explicitely the temporal extent of the coverage</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="DomainExtentType">
                <xs:choice maxOccurs="unbounded">
                    <xs:element name="TimeInterval" type="TimeIntervalType" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:element name="TimeSeries" type="TimeSeriesType" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:element name="defaultTimeInstant" type="xs:dateTime" minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>If multiple temporal coverage, this tag is
required</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                </xs:choice>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="TimeSeriesType">
```

```
    <xs:annotation>
        <xs:documentation>Describe a series of time instant</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="TimeInstant" type="xs:dateTime" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="TimeIntervalType">
    <xs:annotation>
        <xs:documentation>Describe a series of time instant at regular interval</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="minT" type="xs:dateTime"/>
        <xs:element name="maxT" type="xs:dateTime"/>
        <xs:element name="resT" type="xs:duration" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
<!--RangeSet related types and elements-->
<xs:element name="RangeSetDescription" type="RangeSetDescriptionType"/>
<xs:element name="RangeComponentDescription" type="RangeComponentDescriptionType"/>
<xs:complexType name="RangeSetDescriptionType">
    <xs:annotation>
        <xs:documentation>Describe all the component of range set</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element ref="RangeComponentDescription" maxOccurs="unbounded">
            <xs:annotation>
                <xs:documentation>This describes the range  provided by a coverage. </xs:documentation>
            </xs:annotation>
        </xs:element>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="RangeComponentDescriptionType">
    <xs:annotation>
        <xs:documentation>Describe  a range set component</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element ref="RangeID"/>
        <xs:element name="title" type="xs:string"/>
        <xs:element name="description" type="xs:string" minOccurs="0"/>
        <xs:element ref="Observable" minOccurs="0"/>
        <xs:element ref="RangeAxis" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:element name="RangeID" type="xs:string"/>
<xs:complexType name="RangeSubsetType">
    <xs:annotation>
        <xs:documentation>Use for requesting a range subset in GetCoverage</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:choice>
            <xs:sequence maxOccurs="unbounded">
                <xs:element ref="RangeID"/>
                <xs:element name="Parameter" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:complexContent>
                            <xs:extension base="ExtentType">
                                <xs:attribute name="name" type="xs:QName" use="required"/>
                            </xs:extension>
                        </xs:complexContent>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:element name="Parameter" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
```

```
                        <xs:complexContent>
                            <xs:extension base="ExtentType">
                                <xs:attribute name="name" type="xs:QName" use="required"/>
                            </xs:extension>
                        </xs:complexContent>
                    </xs:complexType>
                </xs:element>
            </xs:choice>
        </xs:sequence>
    </xs:complexType>
    <!--Domain set related types-->
    <xs:complexType name="DomainObjectType">
        <xs:annotation>
            <xs:documentation>Describe a spatio temporal object. The spatial and time are separated for the
moment. Temporal needs to be integrated in the geometry in the future. See introduction of ISO
19108</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element ref="gml:_Geometry"/>
            <xs:element ref="xlinks:MultiPolygon"/>
            <xs:element ref="gml:ArcString"/>
            <xs:element name="Elevation" type="ExtentType" minOccurs="0"/>
            <xs:element name="Time" type="ExtentType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <!--Interpolation related types-->
    <xs:element name="SupportedInterpolationList" type="SupportedInterpolationListType"/>
    <xs:element name="InterpolationMethod" type="InterpolationMethodType"/>
    <xs:simpleType name="InterpolationMethodType">
        <xs:annotation>
            <xs:documentation>List of codes that identifies interpolation methods for
coverages.</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:enumeration value="nearest neighbor"/>
            <xs:enumeration value="linear"/>
            <xs:enumeration value="bilinear"/>
            <xs:enumeration value="bicubic"/>
            <xs:enumeration value="lost area"/>
            <xs:enumeration value="barycentric"/>
            <xs:enumeration value="piecewise constant"/>
            <xs:enumeration value="none"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="SupportedInterpolationListType">
        <xs:sequence>
            <xs:element ref="InterpolationMethod" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="default" type="xs:string" use="optional" default="nearest neighbor"/>
    </xs:complexType>
    <!--Grid axis definition-->
    <xs:element name="GridAxis" type="GridAxisType"/>
    <xs:complexType name="GridAxisType">
        <xs:sequence>
            <xs:element name="Name" type="xs:string"/>
            <xs:element name="description" type="xs:string" minOccurs="0"/>
            <xs:element name="orientation">
                <xs:annotation>
                    <xs:documentation>This is the orientation of the grid axis. Natural orientation are right,
up,front</xs:documentation>
                </xs:annotation>
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="up"/>
                        <xs:enumeration value="down"/>
```

```
                        <xs:enumeration value="left"/>
                        <xs:enumeration value="right"/>
                        <xs:enumeration value="back"/>
                        <xs:enumeration value="front"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <!--Common types that needs to be put in a upper schema and be shared by other geospatial data services-->
    <xs:element name="LayerID" type="xs:string"/>
    <xs:element name="KeywordList" type="KeywordListType"/>
    <xs:element name="Keyword" type="KeywordType"/>
    <xs:element name="MetadataURL" type="MetadataURLType"/>
    <xs:element name="SupportedFormatList" type="SupportedFormatListType"/>
    <xs:element name="Format" type="FormatType"/>
    <xs:element name="FormatName" type="xs:string">
        <xs:annotation>
            <xs:documentation>This element provides an identifier for the format in which a layer can be
requested from a service</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="SupportedSRSList" type="SupportedSRSListType"/>
    <xs:element name="SupportedSRS" type="SupportedSRSType"/>
    <xs:element name="LatLonBoundingBox" type="LatLonBoundingBoxType"/>
    <xs:element name="SRS" type="xs:string">
        <xs:annotation>
            <xs:documentation>Use as the identifier of the SRS in the request</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:simpleType name="LayerName">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:complexType name="LayerType">
        <xs:annotation>
            <xs:documentation>This abstract class provides the common attributes for a layer. I could be reused
by WMS, WCS and probably by WFS (despite this notion does not exist in the latest). A layer describes the
underlying data and the processing performed by the service on this data (formatting,
reprojection,...)</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element ref="LayerID"/>
            <xs:element name="Title" type="xs:string"/>
            <xs:element name="Abstract" type="xs:string" minOccurs="0"/>
            <xs:element ref="KeywordList" minOccurs="0"/>
            <xs:element ref="MetadataURL" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="LatLonBoundingBox">
                <xs:annotation>
                    <xs:documentation>This type is provided mainly to define the (approximate if necessary)
long/lat bounding box of the underlying data of a layer. This is mainly used for catalog search </xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:choice>
                <xs:element ref="SRS" maxOccurs="unbounded"/>
                <xs:sequence>
                    <xs:element name="QuerySRS">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="SRS" maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="ResponseSRS">
                        <xs:complexType>
                            <xs:sequence>
```

```
                                <xs:element ref="SRS" maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:choice>
            <xs:element ref="SupportedFormatList"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="LatLonBoundingBoxType">
        <xs:attribute name="minx" type="xs:double" use="required"/>
        <xs:attribute name="miny" type="xs:double" use="required"/>
        <xs:attribute name="maxx" type="xs:double" use="required"/>
        <xs:attribute name="maxy" type="xs:double" use="required"/>
    </xs:complexType>
    <xs:complexType name="KeywordListType">
        <xs:annotation>
            <xs:documentation>A thesaurus can be associated with the list of keywords. It is optional. Mixed of
thesaurus can be done by attributing the thesaurus for each keyword</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element ref="Keyword" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="thesaurus" type="xs:anyURI" use="optional"/>
    </xs:complexType>
    <xs:complexType name="KeywordType">
        <xs:annotation>
            <xs:documentation>A keyword can be associated with a thesaurus (RDF schema) providing the
semantics definition. The value of the keyword will corresponds to the identifier of a ressource</xs:documentation>
        </xs:annotation>
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="thesaurus" type="xs:anyURI" use="optional"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="MetadataURLType">
        <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
        <xs:attribute name="metadataType" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="TC211"/>
                    <xs:enumeration value="FGDC"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
    <xs:complexType name="SupportedFormatListType">
        <xs:sequence>
            <xs:element ref="Format" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="nativeFormat" type="xs:string" use="optional"/>
    </xs:complexType>
    <xs:complexType name="FormatType">
        <xs:annotation>
            <xs:documentation>Describe the format from which the content can be retrieved</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element ref="FormatName"/>
            <xs:element name="description" type="xs:string" minOccurs="0"/>
            <xs:element name="MIMEType" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="SupportedSRSListType">
        <xs:sequence>
```

```
            <xs:element name="SupportedSRS" type="SupportedSRSType" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="nativeSRS" type="xs:string" use="required"/>
    </xs:complexType>
    <xs:complexType name="SupportedSRSType">
        <xs:annotation>
            <xs:documentation>Describe the identifier for the  supported SRS (used by request) and the
associated spatial extent</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element ref="SRS"/>
            <xs:element ref="SpatialExtent"/>
        </xs:sequence>
    </xs:complexType>
    <!--This schema redefines grid geometries from coverage solving some  inconsistencies-->
    <!--Global elements definition-->
    <xs:element name="Grid" type="GridType"/>
    <xs:element name="RectifiedGrid">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="RectifiedGridType">
                    <xs:attribute name="srsName" type="xs:anyURI" use="required"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="RectifiableGrid" type="RectifiableGridType">
        <xs:annotation>
            <xs:documentation>Not fully defined, waiting for MathTransform definition from CT
workgroup</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="GriddedBox" type="SpatialExtentType"/>
    <xs:element name="GridCoord" type="GridCoordType"/>
    <xs:element name="GridCoordinate" type="GridCoordinateType"/>
    <xs:element name="GridRange" type="GridRangeType"/>
    <!--Complex types-->
    <xs:complexType name="GridType">
        <xs:annotation>
            <xs:documentation>Implicitly defines an unrectified grid, which is a network composed of two or more
sets of equally spaced parallel lines in which the members of each set intersect the members of the other sets at
right angles.</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element ref="GridRange"/>
        </xs:sequence>
        <xs:attribute name="dimension" type="xs:positiveInteger" use="required"/>
        <xs:attribute name="type" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="post"/>
                    <xs:enumeration value="centre"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
    <xs:complexType name="RectifiedGridType">
        <xs:annotation>
            <xs:documentation>A rectified grid has an origin and vectors that define its post locations in the
projection defined by the grid coverage</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="GridType">
                <xs:sequence>
                    <xs:element name="origin">
```

```
                        <xs:annotation>
                            <xs:documentation>Origin is the translation to apply to the grid point (0,0) in the SRS
indicated in the RectifiedGrid</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="X" type="xs:double"/>
                                <xs:element name="Y" type="xs:double"/>
                                <xs:element name="Z" type="xs:double" minOccurs="0"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="offsets">
                        <xs:annotation>
                            <xs:documentation>Offset appears in the same order the axis
names</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="offset" minOccurs="2" maxOccurs="3">
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element name="endX" type="xs:double"/>
                                            <xs:element name="endY" type="xs:double"/>
                                            <xs:element name="endZ" type="xs:double" minOccurs="0"/>
                                        </xs:sequence>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="RectifiableGridType">
        <xs:annotation>
            <xs:documentation>This type will be used to handle polynomial, rational
functions,...</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="GridType">
                <xs:sequence>
                    <xs:element name="MathTransform" type="MathTransformType">
                        <xs:annotation>
                            <xs:documentation>Will be defined by CT workgroup in the
future.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="MathTransformType">
        <xs:annotation>
            <xs:documentation>To be defined by CT WG</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="class">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="Polynomial"/>
                        <xs:enumeration value="Rational"/>
                        <xs:enumeration value="Affine"/>
                    </xs:restriction>
```

```
                </xs:simpleType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="GridCoordType">
        <xs:annotation>
            <xs:documentation>Represents a coordinate in a point grid. The coordinates must be integers.The ordering of the coordinate values must be the same as that of the elements of GridAxis.The coordinates values represents the number of offsets from the origin in the direction of a specific axis.</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="ordinate" type="xs:int" minOccurs="2" maxOccurs="4"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="GridCoordinateType">
        <xs:annotation>
            <xs:documentation>Represents a coordinate in a point grid as a string. The coordinates must be integers.The ordering of the coordinate values must be the same as that of the elements of GridAxis.The coordinates values represents the number of offsets from the origin in the direction of a specific axis. Coordinates can be included in a single string, but there is no facility for validating string content. The value of the 'cs' attribute is the separator for coordinate values, and the value of the 'ts' attribute gives the tuple separator (a single space by default); the default values may be changed to reflect local usage.
</xs:documentation>
        </xs:annotation>
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="decimal" type="xs:string" default="."/>
                <xs:attribute name="cs" type="xs:string" default=","/>
                <xs:attribute name="ts" type="xs:string" default="&#x20;"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="GridRangeType">
        <xs:annotation>
            <xs:documentation>Provides grid coordinate values for the diametrically opposed corners of an envelope that bounds a section of grid. The order of the ordinates are the same than the axis described in the GridGeometry</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="low" type="GridCoordType"/>
            <xs:element name="high" type="GridCoordType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="GriddedBoxType">
        <xs:annotation>
            <xs:documentation>Complex type to describe more intuitively a non rotated rectified grid</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="minX" type="xs:double"/>
            <xs:element name="maxX" type="xs:double"/>
            <xs:element name="minY" type="xs:double"/>
            <xs:element name="maxY" type="xs:double"/>
            <xs:element name="minZ" type="xs:double" minOccurs="0"/>
            <xs:element name="maxZ" type="xs:double" minOccurs="0"/>
            <xs:element name="resX">
                <xs:simpleType>
                    <xs:restriction base="xs:double">
                        <xs:minExclusive value="0"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="resY">
                <xs:simpleType>
                    <xs:restriction base="xs:double">
```

```
                        <xs:minExclusive value="0"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="resZ" minOccurs="0">
                <xs:simpleType>
                    <xs:restriction base="xs:double">
                        <xs:minExclusive value="0"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="srsName" type="xs:anyURI" use="required"/>
    </xs:complexType>
    <!-- Grid Band description -->
    <xs:element name="CategoryList" type="CategoryListType"/>
    <xs:element name="Category" type="CategoryType" abstract="true"/>
    <xs:element name="NominalCategory" type="NominalCategoryType" substitutionGroup="Category"/>
    <xs:element name="OrdinalCategory" type="OrdinalCategoryType" substitutionGroup="Category"/>
    <xs:element name="IntervalCategory" type="IntervalCategoryType" substitutionGroup="Category"/>
    <xs:element name="UnitOfMeasure" type="UnitOfMeasureType"/>
    <xs:element name="PseudoColorTable" type="PseudoColorTableType"/>
    <xs:element name="Observable" type="ObservableType"/>
    <xs:element name="Count" type="CountType"/>
    <xs:element name="Histogram" type="HistogramType"/>
    <xs:element name="ColorInterpretation" type="ColorInterpretationType"/>
    <xs:element name="SampleEncoding" type="SampleEncodingType"/>
    <!--Pseudo Color Table definition-->
    <xs:element name="GrayEntry">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="PCTEntryType">
                    <xs:attribute name="gray" use="required">
                        <xs:simpleType>
                            <xs:restriction base="xs:integer">
                                <xs:minInclusive value="0"/>
                                <xs:maxInclusive value="255"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:attribute>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="CMYKEntry">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="PCTEntryType">
                    <xs:attributeGroup ref="CMYK"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="RGBAEntry">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="PCTEntryType">
                    <xs:attributeGroup ref="RGBA"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="HLSEntry">
        <xs:complexType>
            <xs:complexContent>
```

```
                <xs:extension base="PCTEntryType">
                    <xs:attributeGroup ref="HLS"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:complexType name="PseudoColorTableType">
        <xs:choice>
            <xs:sequence>
                <xs:element ref="RGBAEntry" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:sequence>
                <xs:element ref="HLSEntry" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:sequence>
                <xs:element ref="CMYKEntry" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:sequence>
                <xs:element ref="GrayEntry" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:choice>
    </xs:complexType>
    <xs:complexType name="PCTEntryType" abstract="true">
        <xs:attribute name="index" type="xs:integer" use="required"/>
    </xs:complexType>
    <xs:attributeGroup name="RGBA">
        <xs:attribute name="red" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:integer">
                    <xs:minInclusive value="0"/>
                    <xs:maxInclusive value="255"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="green" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:integer">
                    <xs:minInclusive value="0"/>
                    <xs:maxInclusive value="255"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="blue" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:integer">
                    <xs:minInclusive value="0"/>
                    <xs:maxInclusive value="255"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="alpha" use="optional" default="0">
            <xs:simpleType>
                <xs:restriction base="xs:integer">
                    <xs:minInclusive value="0"/>
                    <xs:maxInclusive value="255"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:attributeGroup>
    <xs:attributeGroup name="CMYK">
        <xs:attribute name="cyan" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:integer">
                    <xs:minInclusive value="0"/>
                    <xs:maxInclusive value="255"/>
```

```
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="magenta" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:integer">
                    <xs:minInclusive value="0"/>
                    <xs:maxInclusive value="255"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="yellow" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:integer">
                    <xs:minInclusive value="0"/>
                    <xs:maxInclusive value="255"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="black" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:integer">
                    <xs:minInclusive value="0"/>
                    <xs:maxInclusive value="255"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:attributeGroup>
    <xs:attributeGroup name="HLS">
        <xs:attribute name="hue" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:decimal">
                    <xs:minInclusive value="0"/>
                    <xs:maxInclusive value="360"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="saturation" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:decimal">
                    <xs:minInclusive value="0"/>
                    <xs:maxInclusive value="100"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="lightness" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:decimal">
                    <xs:minInclusive value="0"/>
                    <xs:maxInclusive value="100"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:attributeGroup>
    <!--Histogram definition-->
    <xs:complexType name="HistogramType">
        <xs:sequence>
            <xs:element ref="Count" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="min" type="xs:decimal" use="optional"/>
        <xs:attribute name="max" type="xs:decimal" use="optional"/>
        <xs:attribute name="stddev" type="xs:decimal" use="optional"/>
        <xs:attribute name="mean" type="xs:decimal" use="optional"/>
        <xs:attribute name="median" type="xs:decimal" use="optional"/>
        <xs:attribute name="totalCount" type="xs:nonNegativeInteger" use="required"/>
```

```
    </xs:complexType>
    <xs:complexType name="CountType">
        <xs:attribute name="index" type="xs:decimal" use="required"/>
        <xs:attribute name="count" type="xs:integer" use="required"/>
        <xs:attribute name="percent" use="optional">
            <xs:simpleType>
                <xs:restriction base="xs:decimal">
                    <xs:minInclusive value="0"/>
                    <xs:maxInclusive value="100"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
    <xs:simpleType name="ColorInterpretationType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="redBand"/>
            <xs:enumeration value="greenBand"/>
            <xs:enumeration value="blueBand"/>
            <xs:enumeration value="cyanBand"/>
            <xs:enumeration value="magentaBand"/>
            <xs:enumeration value="yellowBand"/>
            <xs:enumeration value="blackBand"/>
            <xs:enumeration value="hueBand"/>
            <xs:enumeration value="saturationBand"/>
            <xs:enumeration value="lightnessBand"/>
            <xs:enumeration value="alphaBand"/>
            <xs:enumeration value="undefined"/>
            <xs:enumeration value="grayIndex"/>
            <xs:enumeration value="paletteIndex"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="SampleEncodingType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="1BIT"/>
            <xs:enumeration value="2BIT"/>
            <xs:enumeration value="4BIT"/>
            <xs:enumeration value="8BIT_U"/>
            <xs:enumeration value="8BIT_S"/>
            <xs:enumeration value="16BIT_U"/>
            <xs:enumeration value="16BIT_S"/>
            <xs:enumeration value="32BIT_U"/>
            <xs:enumeration value="32BIT_S"/>
            <xs:enumeration value="32BIT_REAL"/>
            <xs:enumeration value="64BIT_REAL"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="ObservableType">
        <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="description" type="xs:string" minOccurs="0"/>
            <xs:element name="docURL" type="xs:anyURI" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>Binds the measurand to a value from a ObservableType registry. This
should be described in a RDF file</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="referenceSystem" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>Binds the measurand to a specific reference system. (can be a unit of
measure, ordinal, nominal categories).</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                    <xs:complexContent>
                        <xs:extension base="ReferenceSystemType">
                            <xs:choice>
```

```
                                <xs:element ref="UnitOfMeasure"/>
                                <xs:element ref="CategoryList"/>
                            </xs:choice>
                        </xs:extension>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="GridRangeDescriptionType">
        <xs:complexContent>
            <xs:extension base="RangeComponentDescriptionType">
                <xs:sequence>
                    <xs:element ref="ColorInterpretation" minOccurs="0"/>
                    <xs:element ref="SampleEncoding" minOccurs="0"/>
                    <xs:element ref="PseudoColorTable" minOccurs="0"/>
                    <xs:element ref="Histogram" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:element name="GridRangeDescription" type="GridRangeDescriptionType"/>
    <!--Reference System definition-->
    <xs:complexType name="ReferenceSystemType"/>
    <xs:complexType name="CategoryType" abstract="true">
        <xs:sequence>
            <xs:element name="categoryID" type="xs:string"/>
            <xs:element ref="title"/>
            <xs:element ref="description" minOccurs="0"/>
            <xs:element name="docURL" type="xs:anyURI" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="NominalCategoryType">
        <xs:complexContent>
            <xs:extension base="CategoryType">
                <xs:sequence>
                    <xs:element name="toNumericValue" type="xs:decimal">
                        <xs:annotation>
                            <xs:documentation>Value used to represent the category. The order is not significant.
</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="OrdinalCategoryType">
        <xs:complexContent>
            <xs:extension base="CategoryType">
                <xs:sequence>
                    <xs:element name="toNumericValue" type="xs:decimal">
                        <xs:annotation>
                            <xs:documentation>Value used to represent the category. The order is
significant</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="IntervalCategoryType">
        <xs:complexContent>
            <xs:extension base="CategoryType">
                <xs:sequence>
                    <xs:element name="lower" type="xs:decimal">
```

```
                        <xs:annotation>
                            <xs:documentation>InclusiveLoweValue</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="upper" type="xs:decimal">
                        <xs:annotation>
                            <xs:documentation>ExclusiveUpperValue</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="toValue" type="ConversionType" minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>conversion from sample to true value</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                </xs:sequence>
                <xs:attribute name="uom" type="xs:anyURI" use="optional"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="ConversionType">
        <xs:choice>
            <xs:element name="affine">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="valueScale" type="xs:decimal"/>
                        <xs:element name="valueOffset" type="xs:decimal"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="logarithmic">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="valueScale" type="xs:decimal"/>
                        <xs:element name="valueOffset" type="xs:decimal"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:choice>
    </xs:complexType>
    <xs:complexType name="CategoryListType">
        <xs:choice>
            <xs:annotation>
                <xs:documentation>Note that mixed category are autorized (oceanography for example use mixed
qualititative and quantitative classification)</xs:documentation>
            </xs:annotation>
            <xs:element ref="NominalCategory" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="OrdinalCategory" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="IntervalCategory" minOccurs="0" maxOccurs="unbounded"/>
        </xs:choice>
    </xs:complexType>
    <xs:complexType name="UnitOfMeasureType">
        <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="shortName" type="xs:string" minOccurs="0"/>
            <xs:element name="description" type="xs:string" minOccurs="0"/>
            <xs:element name="docURL" type="xs:anyURI" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="RangeAxis">
        <xs:annotation>
            <xs:documentation>Set of values AT WHICH obervable is measured, or BY WHICH it is
aggregated</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
```

```
                <xs:extension base="ObservableType">
                    <xs:sequence>
                        <xs:element name="RangeExtent">
                            <xs:annotation>
                                <xs:documentation>Points or intervals along the axis where values are available
(e.g., bands or age brackets)</xs:documentation>
                            </xs:annotation>
                            <xs:complexType>
                                <xs:complexContent>
                                    <xs:extension base="ExtentType">
                                        <xs:sequence minOccurs="0">
                                            <xs:element name="Default" type="xs:string"/>
                                        </xs:sequence>
                                    </xs:extension>
                                </xs:complexContent>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="DescribeCoverageLayer">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="LayerID" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="version" type="xs:string" use="optional" fixed="0.7"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

A.2 Sample WMS Capabilities XML

Here is an example Capabilities XML file that complies with the above Schema document.

(to be inserted)

A.3 Service Exception Schema (Normative)

This annex contains the Service Exception Schema corresponding to this version of the WCS specification. This section also summarizes the defined exception codes and their meanings.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--W3C Schema generated by XML Spy v4.2 U (http://www.xmlspy.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="ServiceExceptionReport">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="ServiceException" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:simpleContent>
                            <xs:extension base="xs:string">
                                <xs:attribute name="code">
                                    <xs:simpleType>
                                        <xs:restriction base="xs:NMTOKEN">
                                            <xs:enumeration value="InvalidFormat"/>
                                            <xs:enumeration value="InvalidSRS"/>
                                            <xs:enumeration value="LayerNotDefined"/>
                                            <xs:enumeration value="CurrentUpdateSequence"/>
                                            <xs:enumeration value="InvalidUpdateSequence"/>
                                            <xs:enumeration value="MissingDimensionValue"/>
                                            <xs:enumeration value="InvalidDimensionValue"/>
                                        </xs:restriction>
                                    </xs:simpleType>
                                </xs:attribute>
                            </xs:extension>
                        </xs:simpleContent>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="version" type="xs:string" fixed="0.5.1"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Table A.1 — Exception codes defined by this specification

Exception Code	Meaning
InvalidFormat	Request contains a Format not offered by the service instance.
InvalidSRS	Request contains an SRS not offered by the service instance for one or more of the Layers in the request.
LayerNotDefined	Request is for a Layer not offered by the service instance.
CurrentUpdateSequence	Value of (optional) UpdateSequence parameter in GetCapabilities request is equal to current value of Capabilities XML update sequence number.
InvalidUpdateSequence	Value of (optional) UpdateSequence parameter in GetCapabilities request is greater than current value of Capabilities XML update sequence number.
MissingDimensionValue	Request does not include a sample dimension value, and the service instance did not declare a default value for that dimension.
InvalidDimensionValue	Request contains an invalid sample dimension value.

A.4 Sample Service Exception XML (Informative)

As an aid to understanding and a guide for implementation, this Annex contains **example** XML which is valid according to the DTD in Annex A.3. Implementers should consult the main body of the specification document and the DTD to ensure compliance rather than editing this XML without full understanding.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<ServiceExceptionReport xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
    xsi:noNamespaceSchemaLocation="ServiceException.xsd" version="0.5.1">
    <ServiceException>
            Plain text message about an error.
    </ServiceException>
    <ServiceException code="InvalidUpdateSequence">
        Another error message, this one with an exception code supplied.
    </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>
```

Annex D
(normative)

Conformance Tests

Bibliography

The OpenGIS Abstract Specification (Topic 6, "The Coverage Type," OGC document #99-106) and the OpenGIS Implementation Specification for Grid Coverages (OGC document #01-004) define coverages, i.e., the spatial function graphs, grids, or data "bundles" returned in response to queries described here.