Open Geospatial Consortium Inc.

Date: 2009-04-08

Reference number of this document: OGC 09-018

Version: **0.2.2**

Category: OpenGIS® Discussion Paper

Editors: Ben Domenico and Stefano Nativi

Web Coverage Service (WCS) 1.1 extension for CF-netCDF 3.0 encoding

Warning

This document is not an OGC Standard. This document is an OGC Discussion Paper and is therefore not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, an OGC Discussion Paper should not be referenced as required or mandatory technology in procurements.

Document type:	OpenGIS® Discussion Paper
Document subtype:	Extension
Document stage:	Draft proposed version 0.2.2
Document language:	English

This page left intentionally blank

Contents

Page

Figures

Page

Tables

Page

Preface

This Discussion Paper describes a draft standard is an extension of the Web Coverage Service (WCS) version 1.1.2 Implementation Standard [OGC 07-067r5]. With small changes, this extension is expected to also apply to WCS 1.2.0. Unlike previous versions, WCS 1.1 and 1.2 are divided into a base standard plus multiple extensions (formerly called application profiles).

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

Document terms and definitions

This document uses the standard terms defined in Subclause 5.3 of [OGC 06-121r3], which is based on the ISO/IEC Directives, Part 2. Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this standard.

Data Model: a data model is a way of thinking about scientific data by applying a data model theory. It is an abstraction that describes how datasets are represented and used. In computer terms, a data model can be thought of as equivalent to an abstract object model in Object Oriented Programming in that an abstract data model describes data objects and what methods can be used on them.

NetCDF-CF: netCDF-CF is a standard for data on complex grids –curvilinear in XY; sigma and density-related in Z; climatological and artificial calendars in T; and heading towards "tile mosaics" and 5D forecast ensembles in the near future.

OPeNDAP: for purposes of this document, OPeNDAP is an approach that allows applications to access remote, time-aggregated collections of netCDF-CF files (virtual datasets – often terabyte sized) through the unaltered netCDF API –as if they were local netCDF files.

Document contributor contact points

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

Name	Organization
Ben Domenico	UCAR/Unidata ben@unidata.ucar.edu
Stefano Nativi	CNR/IMAA nativi@imaa.cnr.it
Ethan Davis	UCAR/Unidata edavi@unidata.ucar.edu
Dominic Lowe	British Atmospheric Data Centre d.lowe@rl.ac.uk
Paolo Mazzetti	CNR/IMAA mazzetti@imaa.cnr.it

Revision history

Date	Release	Editor	Primary clauses modified	Description
4 May 2008	0.9 08-NNN	Ben Domenico	All	First draft
27 May 2008	1.0	Stefano Nativi	All	Second draft
23 June 2008	1.1	Stefano Nativi	All	Third draft
07 July 2008	1.2	Stefano Nativi, Dominic Lowe and Ben Domenico	DescribeCoverage, GetCoverage, MIME types	Revision and integration of comments. NetCDF-CF example
30 August 2008	2.0	Stefano Nativi and Paolo Mazzetti	MIME multipart encoding specification	Document revision. Encoding examples
04 January 2009	2.1	Stefano Nativi		OPeNDAP encoding is added
08 April 2009	2.2	Carl Reed Stefano Nativi	Various	Edits required prior to posting as a DP.

Changes to the OGC Abstract Specification

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

Future work

Improvements in this document are desirable to support changes and additions to CF-netCDF encoding. However, it is important that WCS and CF-netCDF remain "loosely coupled" in the sense that each can change and evolve without having to rewrite the other each time.

In particular, this extension standard encoding profile is limited to regular and warped grids as specified by the current WCS 1.1, but irregular grids are important in the CF-netCDF community and work is underway to expand the CF-netCDF to encompass non-gridded datasets. The current plan is to include these augmentations in subsequent versions of this standard extension.

This specification is written for netCDF 3, but netCDF 4 is now being released. Currently the plan is to submit a separate extension standard for CF-netCDF4 as it becomes more heavily used in the community.

Foreword

This document is an extension of the Web Coverage Service (WCS) version 1.1.2 Implementation Standard [OGC 07-067r5] for the CF-natCDF 3 encoding. With small changes, this extension is expected to also apply to WCS version 1.2. Needed changes for subsequent WCS releases can be incorporated by editing Annex B or by adding another Annex that provides the equivalent information for the latest WCS release. This extension is based on change request [OGC-06-043r4], and supersedes that Discussion Paper. This document does not supersede any other previously approved OGC document

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

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

Introduction

The Web Coverage Service (WCS) supports electronic retrieval of geospatial data as "coverages" – that is, digital geospatial information representing space-varying phenomena. A WCS provides client access to potentially detailed and rich sets of geospatial information in forms that are useful for client-side rendering, multi-valued coverages, and input into scientific models and other clients. The WCS is currently limited to quadrilateral grid coverages, providing information at the grid points, usually with interpolation between these grid points.

This extension of the WCS standard specifies a CF-netCDF3 encoding format option. This is based on the netCDF (network Common Data Form) ver. 3.0 file format using the CF (Climate and Forecast) conventions ver. 1.1

Web Coverage Service (WCS) 1.1 extension for CF-netCDF 3.0 encoding

1 Scope

This extension of the WCS standard specifies an Information Community data model with the related encoding that may optionally be implemented by WCS servers. This extension specification allows clients to evaluate, request and use data encoded in CF-netCDF3 format from a WCS server.

This document is an extension of the Web Coverage Service (WCS) 1.1 Corrigendum 2 (version 1.1.2) Implementation Standard [OGC 07-067r5]. With small changes, this extension is expected to also apply to WCS 1.2.

2 WCS CF-netCDF3 encoding extension Standard overview

This extension Standard specifies a CF-netCDF3 data model with the related binary and XML-based encoding formats in which data may be requested by a WCS client and provided by a WCS server. This extension Standard is an optional implementation by servers. The format is netCDF conforming to the Climate and Forecast (CF) conventions (CF-netCDF3). This standard specifies the CF-netCDF data model mapping onto the WCS data model. In addition, this standard specifies the possible binary and XML-based encoding formats returned by a coverage request expressed through the WCS interface. This document is specific to coverage formats encoded in netCDF 3 using CF 1.1.

NetCDF is a widely-used set of interfaces for array-oriented data access and a freely-distributed collection of data access libraries for C, Fortran, C++, Java, and other languages. The netCDF libraries support a machine-independent, self-documenting binary format for representing scientific data. Together, the interfaces, libraries, and format support the creation, access, and sharing of scientific data. The CF conventions define metadata (internal to a netCDF file) that provide a definitive description of what the data in each variable represents, and of the spatial and temporal properties of the data.. This enables the users of data from different sources to decide which quantities are comparable and how they relate to one another in space and time.The perceived need for a WCS 1.1. standard extension for CF-netCDF3 arose from the experiences of the OGC GALEON (Geo-interface for Air Land Environment Ocean Netcdf) Interoperability Experiment. In this experiment, several WCS 1.0 clients were successful in accessing data from WCS 1.0 servers which encoded the data in CF-compliant netCDF (version 3.0) form. The GALEON experiment has proven that the CF-netCDF3 is a viable and valuable WCS encoding format. However, CF-netCDF3 was not among the list of 5 "required supported formats" in the WCS 1.0 specification.

Based on the OGC GALEON experiment and subsequent discussion in the WCS Revision Working Group (RWG) and subsequent Standards Working Group (SWG), this proposal in conjunction with the later WCS 1.x specifications will establish CF-netCDF3 as a WCS "supported output format."

3 Summary of WCS Extensions

This extension specifies the following set of modifications to WCS operations:

- `GetCapabilities` and `DescribeCoverage` for CF-netCDF3 datsets
    - Mapping of CF-netCDF3 data model onto WCS 1.1.2
    - Introduction of a new MIME type: *application/x-netcdf*.
- `GetCoverage` for CF-netCDF datsets
    - Specification of the abstract data model of the WCS GetCoverage response for CF-netCDF datasets.
    - Specification of the content model of the WCS GetCoverage response for the following CF-netCDF datasets encodings:
        - Local binary file format
        - Remote binary file format using OPeNDAP
        - ncML (netCDF Markup Language)
    - Introduction of a new MIME type: *application/x-netcdf*.

4 Content and organization

It is important to note that this document does not contain inline a specification of the netCDF file format or application programming interface. Nor does it contain inline a specification of the CF conventions. The underlying assumption is that parties interested in working with coverages encoded in CF-netCDF3 will do so using existing libraries rather than coding from scratch to the existing specifications, so one goal of this document is to point them to the documentation, code, and related materials they will need to do so.

Hence, this document includes the encoding profile information specified in Section 9.3.2.2 of WCS 1.1 (07-067r2), namely:

a) MIME type(s) and brief description: a concise overview of the encoding format, including the MIME type string(s) used to refer to it, the files required (e.g. header, dictionary, georeferencing, etc.), and the "role" values in the Xlink references in the GetCoverage response (see Subclauses 10.3.11.2 and I.3.2).

b) Pointers to documentation for the encoding format. This documentation shall clarify how the encoding convention represents locations, times, and physical quantities represented in the dataset.

c) Data model mapping to the ISO 19123: Coverage Abstract Specification. This includes conventions for representing the spatio-temporal domain, and for representing the dimensions in the range. It also describes limitations of the format for encoding complex coverages, and limitations of the coverage model for representing complex data structures encoded in the format.

d) Examples: A set of examples of the encoding format, and of corresponding Coverages XML response documents (see Appendix C).

e) Pointers to implementing software for the encoding format. Providers of this software may license it in source code or executable form.

f) Compliance Testing: Pointers to mechanisms for testing whether resulting WCS coverages conform to the encoding format (see Appendix D).

5 Overview of netCDF and CF conventions (Section B)

5.1 NetCDF

NetCDF (network Common Data Form) is an interface for array-oriented data access and a library that provides an implementation of the interface. The netCDF library also defines a machine-independent format for representing scientific data. Together, the interface, library, and format support the creation, access, and sharing of scientific data.

The netCDF software includes C, Fortran 77, Fortran 90, and C++ interfaces for accessing netCDF data. These libraries are available for many common computing platforms.

5.1.1 NetCDF-3 Data Model

The netCDF-3 data model is a simple one made up of four basic pieces: datasets, variables, dimensions, and attributes. The dataset or file itself can contain variables, dimensions, and attributes. Each variable is an n-dimensional array of one of the following types: byte, char, short, int, float, and double. The dimensions are named and are used to define the shape of the variables. Attributes are name/value pairs that can be contained at the dataset or variable level.

The fact that dimensions are named and scoped to the entire dataset is significant in that if the same dimension is used in multiple variables it means that those variables are related in that dimension.

The netCDF3 data model using UML is depicted in Figure 1.

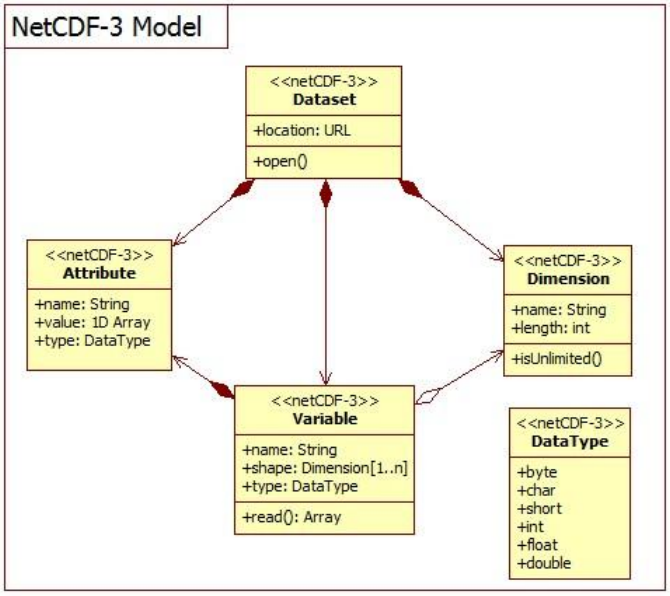

Figure 1 - NetCDF3 data model

5.1.2 NetCDF Coordinate Variables

Variables with a single dimension whose names match the name of their dimension are called “coordinate variables”. Though these coordinate variables have no special meaning in the netCDF API, by convention coordinate variables define the physical coordinate for that dimension (see the Netcdf Users Guide [NUG]). For instance, the following dataset description (in CDL notation, described in the [NUG]) shows a temperature variable that is mapped into latitude and longitude by the lat and lon coordinate variables:

```
netcdf temperature {

  dimensions:
    lat = 45;
    lon = 57;

  variables:
    double lat(lat);
    double lon(lon);

    double temperature(lon, lat);
}
```

5.1.3 NetCDF Standard Attribute Conventions

The [NUG] Appendix B "Attribute Conventions" defines a convention for a number of standard attributes. The convention includes attributes to specify the units of a variable, the minimum and maximum allowed values for a variable, a value to be considered "missing data", scale/offset values, as well as the conventions used in the dataset. Adding some of these conventions to the above example gives us:

```
netcdf temperature {
  dimensions:
    lat = 45;
    lon = 57;

  variables:
    double lat(lat);
      lat:long_name="latitude";
      lat:units="degrees_north";
    double lon(lon);
      lon:long_name="longitude";
      lon:units="degrees_east";

    double temperature(lon, lat);
      temperature:long_name="temperature";
      temperature:units="K";
}
```

5.1.4 NetCDF-3 Binary File Format

From the "NetCDF Users Guide" [NUG] section 4 "File Structure and Performance",

> A netCDF classic or 64-bit offset dataset is stored as a single file comprising two parts:
>
> 1. a header, containing all the information about dimensions, attributes, and variables except for the variable data;

2. a data part, comprising fixed-size data, containing the data for variables that don't have an unlimited dimension; and variable-size data, containing the data for variables that have an unlimited dimension.

Both the header and data parts are represented in a machine-independent form. This form is very similar to XDR (eXternal Data Representation), extended to support efficient storage of arrays of non-byte data.

More detailed information is available in the "File Format Specification" [NUG, appendix C].

5.1.4.1 MIME type (Normative) (Section A)

[Clause] MIME type string used to refer to it:

Note that the only MIME type in use presently is "application/x-netcdf" and it is not officially registered. Thus the MIME types discussed below are new and application for registration will be made to IANA.

binaryapplication/CF-netCDF3

5.1.5 NcML

[Clause] MIME type string used to refer to it:

application/ncML+xml

This xml media type is unregistered.

5.1.6 NetCDF Documenation

The netCDF home page is located at Unidata:

http://www.unidata.ucar.edu/software/netcdf/

A list of netCDF documents is available at:

http://www.unidata.ucar.edu/software/netcdf/docs/

There is a netCDF Users Guide at:

http://www.unidata.ucar.edu/software/netcdf/docs/netcdf.html

The UNIDATA documentation also provides the File Format Specification

http://www.unidata.ucar.edu/packages/netcdf/guide_15.html

for netCDF and Best Practice recommendations on writing netCDF files

http://www.unidata.ucar.edu/packages/netcdf/BestPractices.html

The netCDF XML encoding (ncML) is documented at:

http://www.unidata.ucar.edu/software/netcdf/ncml/

http://www.unidata.ucar.edu/projects/THREDDS/BenStuff/Documents/WCSnetCDF.htm

http://linkinghub.elsevier.com/retrieve/pii/S0098300405001019

http://www.unidata.ucar.edu/software/netcdf/ncml/v2.2/AnnotatedSchema.html

5.2 CF Conventions

The CF conventions define metadata that provide a definitive description of what the data in each variable represents, and of the spatial and temporal properties of the data. This enables users of data from different sources to decide which quantities are comparable, and facilitates building applications with powerful extraction, regridding, and display capabilities.

To identify that the file uses the CF convention, it recommends the `Conventions` attribute be given the string value of "CF-1.1".

The CF Conventions home page is located at PCMDI:

http://cf-pcmdi.llnl.gov/

A CF-netCDF data model schema is shown in Figure 2.

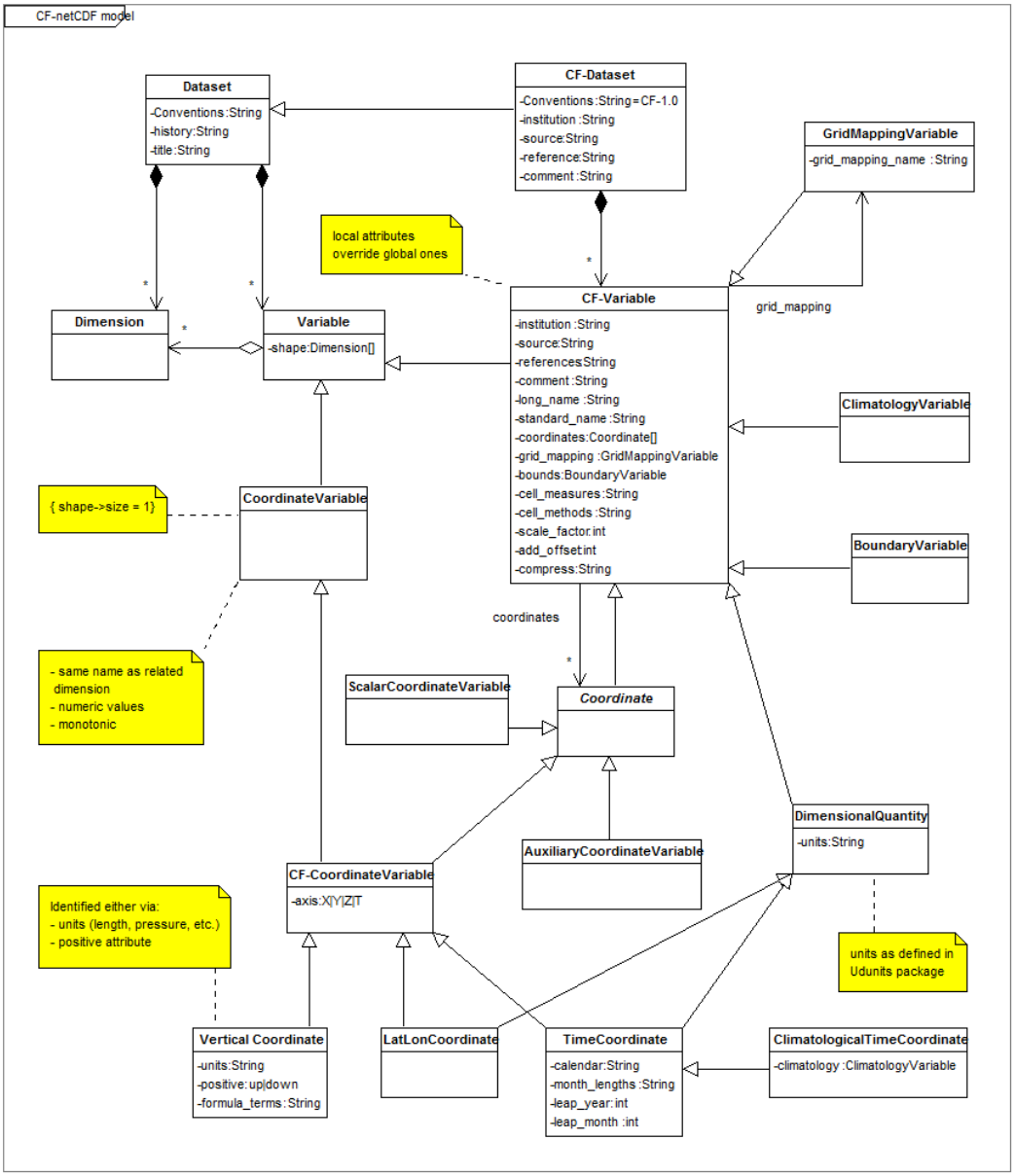

Figure 2 - CF-netCDF data model

5.2.1 CF Standard names

What do the numbers in a netCDF dataset represent? e.g., temperature, pressure, wind speed, salinity, radiance, reflectivity.

Since netCDF variable names are not always enough to fully describe the physical quantity being represented, the CF convention provides an optional variable attribute, `standard_name`, for associating the variable with a standard name. The CF convention maintains a controlled list of permissible standard names. More information is available from the CF Conventions specification [CF].

- Description of CF standard names conventions: http://cf-pcmdi.llnl.gov/documents/cf-conventions/1.1/ch03s03.html
- Table of CF Standard names: http://cf-pcmdi.llnl.gov/documents/cf-standard-names/7/cf-standard-name-table.html

5.2.2 CF Units

What are the *units of measure* for the numbers in a netCDF dataset?

The CF convention requires the `units` attribute for all variables that represent dimensional quantities. The value of the `units` attribute must be a string that can be recognized by Unidata's Udunits software package [UDUNITS]. More information on the `units` attribute can be found in Section 3.1 "Units" of the CF convention specification [CF].

Description of CF units of measure: http://cf-pcmdi.llnl.gov/documents/cf-conventions/1.1/ch03.html#units

5.2.3 CF Coordinate types and coordinate systems

Where in space do the numbers represent measurements or modeled values?

The CF convention gives special treatment to latitude, longitude, vertical, and time coordinates. "Chapter 4 Coordinate Types" of the CF specification [CF] details how coordinates are identified for these special coordinate types.

"Chapter 5 Coordinate Systems" [CF] explains two methods by which coordinate variables are associated with individual variables to form the coordinate system for that variable. The first method is the same as the netCDF coordinate variable convention defined in the [NUG].

For horizontal coordinates that are not latitude/longitude, section 5.6 "Grid Mappings and Projections" describes how various projections and mappings can be specified. For instance, the following is the previous example in a Transverse Mercator projection:

```
netcdf temperature {
  dimensions:
    x = 45;
    y = 57;

  variables:
    double lat(y, x);
      lat:long_name = ”latitude”;
      lat:units = ”degrees_north”;
    double lon(y, x);
      lon:long_name = ”longitude”;
```

```
      lon:units = ”degrees_east”;

    double temperature(y, x);
      temperature:long_name = ”temperature”;
      temperature:units = ”K”;
      temperature:coordinates = “lat lon”;
      temperature:grid_mapping = “tm_mapping”;

    int tm_mapping;
      tm_mapping:long_name = “British National Grid / OSGB 1936”;
      tm_mapping:grid_mapping_name  = “transverse_mercator”;
      tm_mapping:semi_major_axis    = 6377563.396;
      tm_mapping:inverse_flattening = 299.3249646;
      tm_mapping:latitude_of_projection_origin  = 49.0;
      tm_mapping:longitude_of_projection_origin = -2.0;
      tm_mapping:false_easting      = 400000.0;
      tm_mapping:false_northing     = -100000.0;
      tm_mapping:scale_factor_at_projection_origin
                                    = 0.9996012717;
}
```

Coordinate types description:
http://cf-pcmdi.llnl.gov/documents/cf-conventions/1.1/ch04.html

Coordinate systems description:
http://cf-pcmdi.llnl.gov/documents/cf-conventions/1.1/ch05.html

5.2.4 CF Grid Cells

The CF convention defines a number of methods for defining the extent of the grid cells in “Chapter 7 Data Representative of Cells” [CF]. For instance, a `bounds` attribute can be added to appropriate coordinate variables. The `bounds` value is the name of the variable containing the vertices of the cell boundaries.

```
  dimensions:
    lat = 45;
    lon = 57;
    nv = 2; // number of vertices

  variables:
    double lat(lat);
      lat:long_name = ”latitude”;
      lat:units = ”degrees_north”;
      lat:bounds = “lat_bnds”;
    double lon(lon);
      lon:long_name = ”longitude”;
      lon:units = ”degrees_east”;
      lon:bounds = “lon_bnds”;

    double lat_bnds(lat, nv);
```

```
    double lon_bnds(lon, nv);
```

Grid mappings description:
http://cf-pcmdi.llnl.gov/documents/cf-conventions/1.1/ch05s06.html

5.2.4.1.1 CF Time coordinate

When were the measurements taken or modeled values forecast?

The CF convention defines a mechanism for specifying information about time and calendars including support for concepts such as 360-day calendars which are used in climate modeling.

Time coordinate description:
http://cf-pcmdi.llnl.gov/documents/cf-conventions/1.1/ch04s04.html

6 Code for Implementing the netCDF Interface (Section E)

The vast majority of netCDF users use the same set of supported netCDF libraries to implement their systems. Consequently anyone intending to access data in netCDF form via the WCS interface should consider using this code which can be downloaded at:

http://www.unidata.ucar.edu/software/netcdf/

Implementations are available in several programming languages:

- Installation instructions for C, Fortran, and C++ libraries
- NetCDF for Java
- Other interfaces to netCDF data:
  - MATLAB:
    http://www.unidata.ucar.edu/software/netcdf/software.html#MATLAB
  - Objective-C:
    http://www.unidata.ucar.edu/software/netcdf/software.html#Objective-C
  - Perl: http://www.unidata.ucar.edu/software/netcdf/software.html#Perl
  - Python:
    http://www.unidata.ucar.edu/software/netcdf/software.html#Python
  - R: http://www.unidata.ucar.edu/software/netcdf/software.html#R
  - Ruby: http://www.unidata.ucar.edu/software/netcdf/software.html#Ruby
  - Tcl/Tk:
    http://www.unidata.ucar.edu/software/netcdf/software.html#Tcl/Tk

7 Support

Potential users of netCDF may be interested in what support is available for the code and interface. Pointers to netCDF FAQ, mailing lists, documentation are available at:

http://www.unidata.ucar.edu/software/netcdf/

8 NetCDF Examples (Section D)

A set of representative examples of netCDF datasets is essential for gaining an understanding. A good place to start is:

http://www.unidata.ucar.edu/software/netcdf/examples/files.html

http://mst.nerc.ac.uk/file_format_netcdf.html

http://badc.nerc.ac.uk/help/formats/netcdf/index_cf.html

9 Compliance (Section F)

Ensuring adherence to WCS is an issue separate from the CF-netCDF encoding. Several documents are available with recommendations for CF-netCDF verification. They are listed in

CF Requirements and Recommendations
http://cf-pcmdi.llnl.gov/conformance/requirements-and-recommendations/

In order to test whether the netCDF encoded file transferred via a WCS getCoverage request complies with CF conventions, a CF conventions compliance test is available at:

CF-netCDF Compliance:
http://badc.nerc.ac.uk/help/formats/netcdf/index_cf.html

10 OPeNDAP

OPeNDAP (Open-source Project for a Network Data Access Protocol)[1] provides software which makes local data accessible to remote locations regardless of local storage format. OPeNDAP software is freely available.

NetCDF Community applications commonly make use of the OPeNDAP approach to access remote, time-aggregated collections of netCDF-CF files (virtual datasets – often terabyte sized) through the unaltered netCDF API –as if they were local netCDF files.

For the scope of this document, a simplified view of OPeNDAP is that it is a transparent mechanism by which an application can use netCDF API calls on a remote file. Thus for

[1] http://opendap.org/

any netCDF subset that may be derived from a WCS server, there may be an OPeNDAP URL that is an indirect reference to that same subset.

For example let us consider the dataset coads_climatology.nc, served by the WCS server implemented by the TDS (THREDDS Data Server) at http://ferret.pmel.noaa.gov/thredds/dodsC/data/PMEL/coads_climatology.nc.html.

The dataset contains 12 months of grids for seven different surface met fields. Imagine a WCS GetCoverage request that would return a netCDF file containing global SST for the month of January. The contents of this exact netCDF subset can be expressed by the OPeNDAP URL: "http://ferret.pmel.noaa.gov/thredds/dodsC/data/PMEL/coads_climatology.nc?COADSX,COADSY,TIME[0:1:0],SST[0:1:0]".

Essentially, the netCDF file is just a de-referencing of this URL. Any application program that can utilize a netCDF file can (in principal) utilize the URL equivalently.

11 CF-netCDF Mapping to WCS Data Model (Normative) (Section C)

For use in the context of WCS, it is important to have an understanding of the mapping between the data models used for OGC coverages and those for the netCDF with CF conventions. What follows is a conceptual overview section for background followed by a more formal and detailed section with UML diagrams comparing the two data models.

The general mapping strategy is depicted in Figure 3

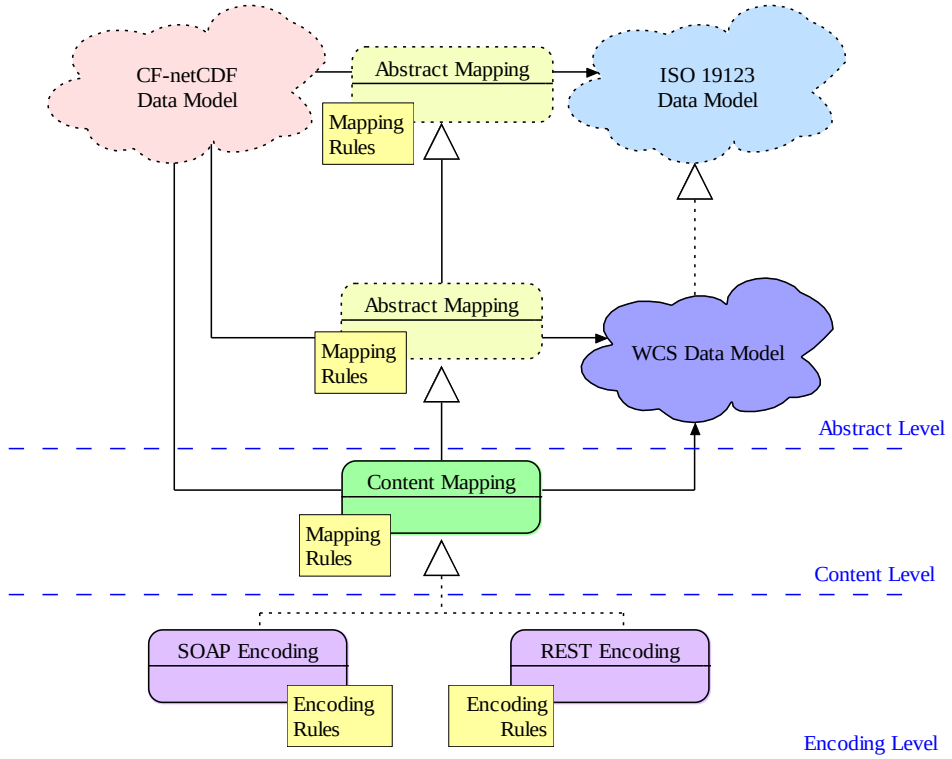

Figure 3. The general strategy followed for the CF-netCDF mapping to WCS Data Model

11.1 NetCDF grid data mapping to ISO Discrete Grid Point Coverage type

Grid is defined as a network composed of two or more sets of curves in which the members of each set intersect the members of the other sets in an algorithmic way. These curves partition a space into grid cells. The axes of the grid provide a basis for defining grid coordinates. The axes need to be identified to support sequencing rules for associating feature attribute value records to the grid points. There are grid points at all grid line intersections; they represent the domain elements. Thus, netCDF gridded data may be effectively mapped onto Discrete Point Coverages whose domain consists of the

point objects characterizing the grid tessellation. The domain of a Discrete Grid Point Coverage instance is a set of Grid Points that are associated with records of feature attribute values through a CV_GridValuesMatrix element.

Therefore, netCDF grid data type must be mapped onto the ISO 19123 CV_DiscreteGridPointCoverage type.

11.1.1 NetCDF Discrete Vs Continuous Coverage types

In most cases, a continuous coverage is also associated with a discrete coverage that provides a set of control values to be used as a basis for evaluating the continuous coverage. Evaluation of the continuous coverage at other direct positions is done by interpolating between the geometry value pairs of the control set. This often depends upon additional geometric objects constructed from those in the control set; these additional objects are typically of higher topological dimension than the control objects.

In ISO 19123, such objects are called "geometry value objects". A geometry value object is a geometric object associated with a set of geometry value pairs that provide the control for constructing the geometric object and for evaluating the coverage at direct positions within the geometric object.

A common example of geometry value object is represented by quadrilateral grid cell whose vertices are represented by four grid points (i.e. the set of geometry value pairs).

In the netCDF domain, the continuous quadrilateral grid coverage type is associated to a discrete grid point coverage type by sharing the same geometry grid and matrix values; the two coverage subclasses share the GridValueMatrix object and the derived GridPointValuePair objects. The real difference consists in the realization of the *locate()* operation, which is inherited from the Coverage super-type. Therefore, "the principal use of discrete point coverages is to provide a basis for continuous coverage functions, where the evaluation of the continuous coverage function is accomplished by interpolation between the points of the discrete point coverage".

In the case of netCDF data, the interpolation methods specified by the ISO continuous coverage classes do not apply in general. In fact, in most cases, any scientifically realistic interpolation depends on the physics of the situation as well as the geometry. Hence, any realistic interpolation is actually data dependent.

Therefore the netCDF gridded data types don't implement the evaluation operation using interpolation methods. They are mapped to the ISO discrete coverages because they actually represent sampled points in a continuous space where the intermediate values depend on the solution to physics-based equations that depend on the values of the range data.

11.2 The Mapping approach

11.2.1 General aproach

Any group of CF-netCDF data variables that share the same set of spatial/temporal coordinate variables can be mapped to a single grid coverage, CV_DescreteGridPointCoverage.

The set of spatial/temporal coordinate variables maps to the domain of the coverage, whose geometry is represented by a single grid, CV_Grid. The grid mapping and/or projection information maps to the CRS and the associated CoordinateSystem (e.g. the units of the coordinate system).

The data variables in the CF-netCDF coverage make up the range with each variable being a separate range field.

Since the coordinate variables do not define the location of the data points with an origin and offset vectors but rather provide a lookup table of sorts, a CF-netCDF coverage can be more closely modeled with a CV_ReferencableGrid than with a CV_RectifiedGrid. However, when the associated coordinate variables are evenly spaced a CF-netCDF coverage can be modeled as a CV_RectifiedGrid. The CF grid cell methods map to CV_GridCell.

11.2.2 Formal approach

To explicitly map the CF-netCDF grid data model (e.g. FES hyperspatial observation and model outputs) to the ISO Coverage data model (i.e. CV_DiscreteGridPointCoverage type), there is a need to address structural and semantics differences, applying the appropriate constraints and, hence, performing a mediation process.

Points to consider include:

- CF-netCDF grid data model supports datasets characterized by multiple domains (e.g. more than one coordinate system is defined for a dataset), whereas an ISO coverage is characterized by a single coordinate system.
- CF-netCDF grid data model supports datasets characterized by arbitrary multi-dimensional domains, whereas an ISO coverage domain is either 2-D (space), 3-D (2D + vertical dimension or 2D + time), 4-D (2D + vertical dimension + time).
- Most commonly, the grid axes of a CF-netCDF dataset coincide with reference system axes. However, CF-netCDF allows arbitrary domain shapes, i.e. grid axes ordering. Thus, it is possible to have a variable v1 defined on a grid <x, y, t, z> and a variable v2 defined on a grid <z, x, t, y>. Since there is a fixed enumeration of allowed compound CRSs in ISO coverages, the transformation of such generalized grids coordinates to CRS coordinates may not be an affine

transformation. In other words, mapping CDM grids to ISO (geo)rectified grids may require axes reshaping and reordering.

There are two main steps to address these mediation issues: a first step consists in defining appropriate profiles for both CF-netCDF and ISO coverage data models, as far as grid point coverage is concerned. The second step deals with defining a set of mapping constraint rules.

11.2.3 CF-netCDF grid data profile model

Figure 4 shows the CF-netCDF profile model. Several CF convention features have been neglected to simplify and generalize the mapping.

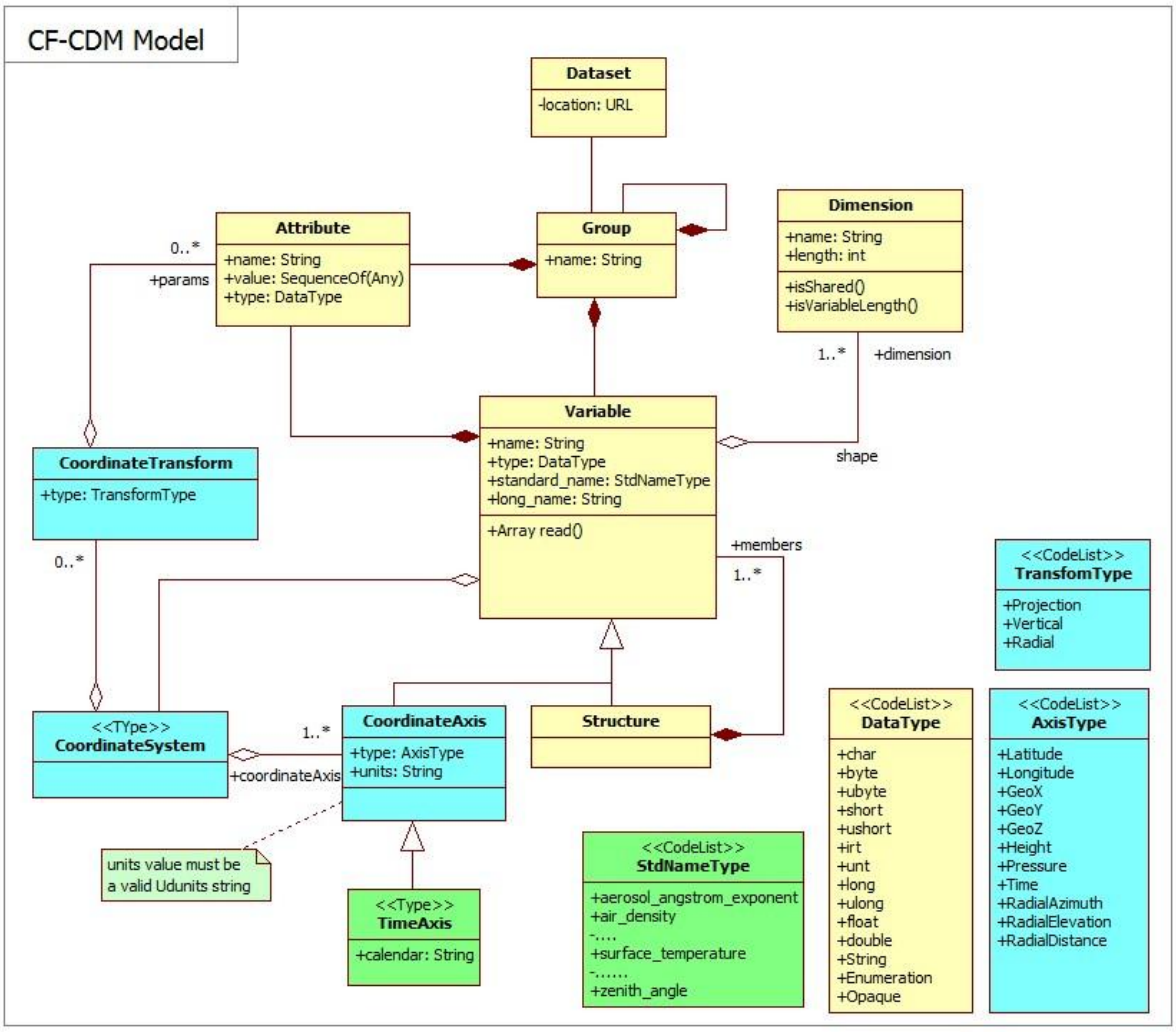

Figure 4 - CF-netCDF profile model

11.2.4 ISO DiscreteGridPointCoverage profile model

The present extension models only regularly spaced domains, adopting the following solution to describe and formalize the domain of discrete grid point coverages:

To implement a RectifiedGrid object (see Figure 5) and its valuation GridValueMatrix object: useful to model only regularly spaced domains

The specific DiscreteGridPointCoverage profile considered by this assumption is shown in Figure 5.

This profile can be used to model also "engineering" grid data: datasets which are referred to a local defined coordinate system. In this case, the CRS is an application defined (i.e. engineering) CRS, characterized by an "engineering" datum (see ISO 19111). The associated Coordinate System have the origin and offset attributes coincident with the data grid Cartesian system origin and axes versors.

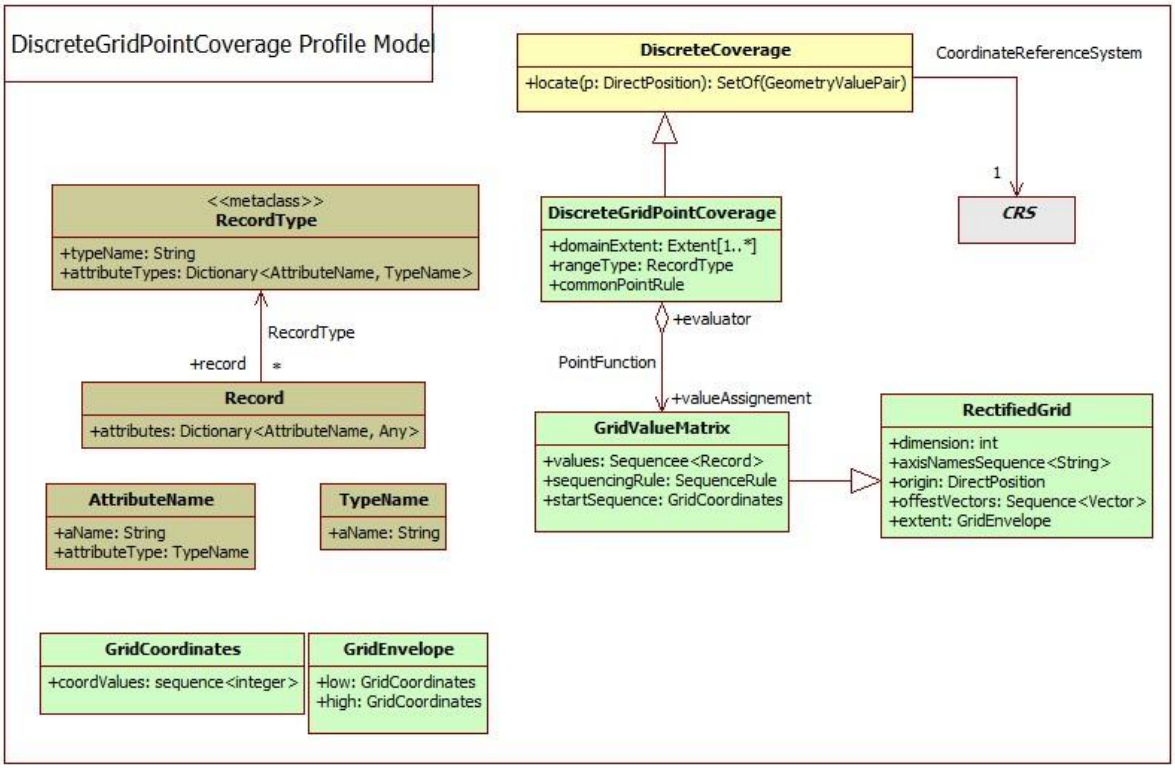

Figure 5 - ISO DiscreteGridPointCoverage profile model

11.2.5 Mapping Rules (Normative)

The high level mapping schema is depicted in Figure 6. The schema shows the CF-netCDF elements to be used in order to generate the ISO 19123 elements which are implemented by the WCS specification.

The mapping rules are reported in the Table 1, Table 2 and Table 3; they are normative. Rules are expressed as relationships between model concepts and a set of constraints; we selected this approach to keep the mapping explanation as simple as possible.

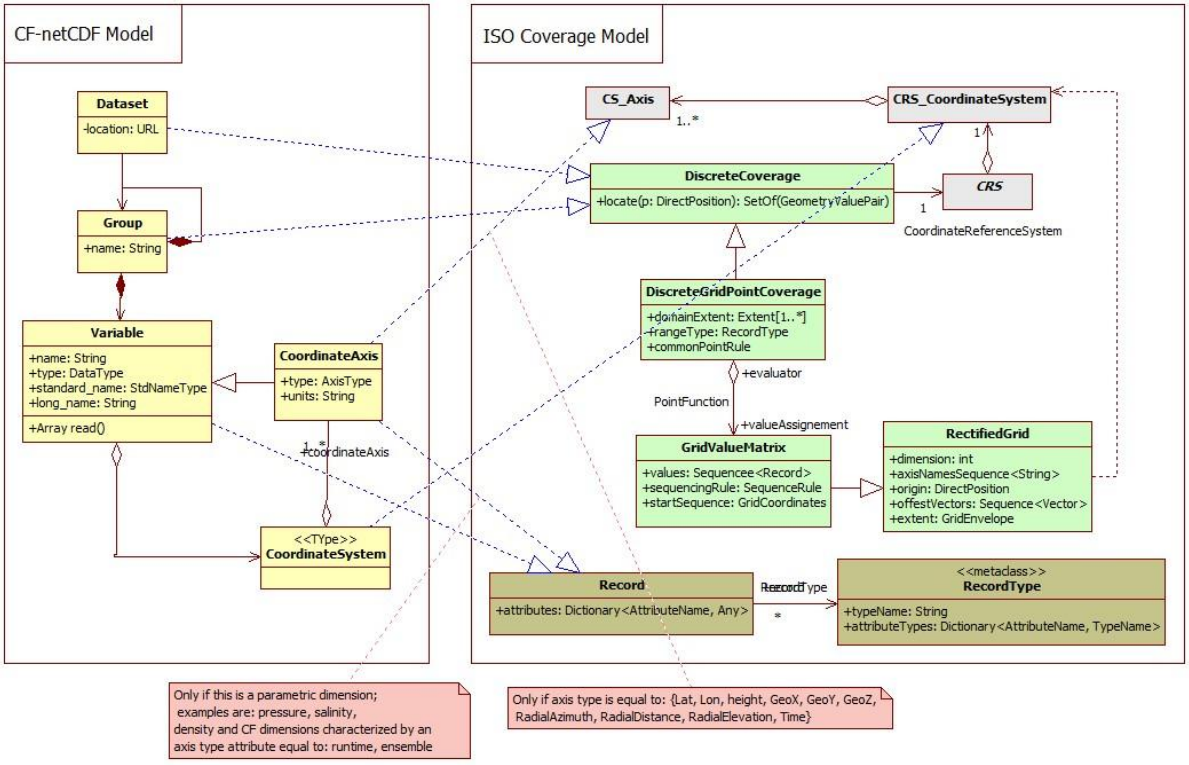

Figure 6 - Mapping Schema

Table 1. Summary of relationship between the CF-netCDF and the DiscreteGridPointCoverage profile models: main packages

CF-netCDF profile concept	ISO Grid Coverage profile concept	Mapping Cardinality (obligation)	Mapping Details
Dataset	DiscreteGridPointCoverage	1 to 0..n	See table Table 2
Coordinate System	CRS.CoordinateSystem	1 to 0..1	See table Table 3
Variable	RangeType entry	1 to 0..1	See table Table 2

11.2.5.1 Dataset and Variables

Table 2. Summary of relationship between the CF-netCDF and the DiscreteGridPointCoverage profile models: Dataset package

CF-netCDF profile concept	ISO Grid Coverage profile concept	Mapping Cardinality	Constraints
Dataset.Group	DiscreteGridPointCoverage	1 to 0..1	Grouping the Variables defined in a dataset by their CoordinateSystem, a DiscreteGridPointCoverage may be defined for each group. The association of groups and coverages may not be one-to-one, since the concept of coordinate system in CF-netCDF is wider than CRS (see table Table 3). Hence, some of the obtained coverages may be further grouped together. It is possible that a CoordinateSystem entity does not contain any axes allowed in coverage CRS (i.e. only parametric dimension axes); the associated variables would then originate no DiscreteGridPointCoverage instance.
Dataset.Group .Variable	DiscreteGridPointCoverage .RangeType.AttributesType entry (i.e. AttributeName, TypeName)	1 to 0..1	The range-set of each DiscreteGridPointCoverage is a list of records with an attribute for every related CF-CDM variable and for every CoordinateAxis of the CoordinateSystem that is not allowed in CRS (i.e. parametric dimension axes)
Dataset.Group .Variable values	GridValueMatrix.values.record entry (i.e. AttributeName, Any)	1 to 0..n	The CF-CDM variable data values generate the grid value matrix record values.

11.2.5.2 Coordinate System and Grid Geometry

Table 3. Summary of relationship between the CF-netCDF and the DiscreteGridPointCoverage profile models: Coordinate System package

CF-netCDF profile concept	ISO Grid Coverage profile concept	Mapping Cardinality	Constraints
CoordinateSystem.CoordinateAxis (space dimension: axis type attribute equal to: *Lat*, *Lon*, *Height*, *GeoX*, *GeoY*, *GeoZ*, *RadialAzimuth*, *RadialElevation*, *RadialDistance*)	CRS.CoordinateSystem.CoordinateSystemAxis	1 to 1	Parametric coordinate systems are allowed in CF-CDM but not in CRS[2]. A coordinate system is of type parametric if a physical or material property is used as a dimension [21]; valuable examples are pressure in meteorology and density in in oceanography. It is possible that a CoordinateSystem entity does not contain any axes allowed in coverage CRS (i.e. only parametric dimension axes); Only spatial and temporal coordinates in a CF-CDM CoordinateSystem become part of a coverage CRS, whereas parametric dimension axes are mapped to compound range set components. The domain of each coverage may be described by the extent of the related coordinate axis variables (if

[2] Future extension to ISO 19111 (see ISO/CD19111-2) may permit parametric CRS, that would accommodate the pressure axis.

			present). The grid geometry of a DiscreteGridPointCoverage may be slanted, with respect to the CRS axes, by specifying appropriate offset vectors. However, the selected CF-netCDF profile permits orthogonal grids only: grids which are aligned with the CoordinateSystem axes.
CoordinateSystem.TimeAxis (axis type attribute equal to: *time*)	CRS.CoordinateSystem.CoordinateSystemAxis	1 to 1	See above
CoordinateSystem.CoordinateAxis (parametric dimension; examples are: pressure, salinity, density and CF dimensions characterized by an axis type attribute equal to: *runtime*, *ensemble*).	DiscreteGridPointCoverage.RangeType.Record.Attributes	1 to 0..1	See above

12 Limitations and future work

As noted in the GALEON IE, many of the difficulties in serving CF netCDF data via WCS 1.0 relate to the fact that, in effect, there are many coverages within one CF-compliant netCDF dataset. Some of these limitations remain in WCS 1.1. Since later versions of WCS are still under revision, the issues are listed here:

- In GALEON phase 1, there was a very simple WCS use case, namely, the client specifies a "parameter" name, a bounding box, and time constraint and gets back a binary "coverage" that, if written to disk, is actually a CF-netCDF3 file containing the requested data subset. This use case represents the real core of WCS for the climate science community. In the 1.1 version of WCS this use case is not

currently possible as a file must always be accompanied with an XML manifest. At the time of writing change request has been submitted [OGC document 07-096] to request that this simple use case is still possible in subsequent versions of WCS.

- NetCDF domain grids can be irregular; but WCS 1.1 provides only for regular and warped grids. Hence, for WCS 1.1, NetCDF grids are limited to those that a WCS 1.1 (actually a GML Coordinate Reference System Transformation) can describe.
- CF-netCDF3 datasets often include data for multiple parameters (e.g. temperature, pressure, wind speed, etc.). Sometimes these different fields have different domains which remains a problem in WCS 1.1
- CF-netCDF3 datasets that contain the output of numerical forecast models in effect have multple time dimensions (the time the model was run and the set of forecast times.) Here again, mechanisms for dealing with this are still limited in WCS 1.1.
- In general, many CF-netCDF3 datasets can be thought of as containing a set of coverages that correspond to multiple coverages (in the traditional sense of a 2D "layer"). For example a CF-netCDF3 dataset may contain multiple "parameters," multiple vertical layers and multiple times. Also, non-traditional (in GIS terms) views of the data may be required (e.g., Hovmoller diagrams where longitude is the x-axis and time is the y-axis)
- In many netCDF datasets, the "height" dimension is represented by a non-spatial coordinate (e.g., pressure in the atmosphere or density in the oceans.) This is still not explicitly possible as a means for representing one of the domain dimensions in WCS 1.1. The main implication is that the bounding vertical limits in a bounding box cannot be expressed in the native elevation dimension in such cases for the netCDF. But a co-domain axis can be defined using the non-spatial dimension.
- An issue that has arisen in GALEON is to include non-gridded coverages such as:
    - collections of observational data taken at points (e.g., river gage or weather observation stations);
    - trajectories such as vertical profiles in the atmosphere or ocean, and collections of such trajectories;
    - time series of measurements at points or of "fields";

13 References

- Proposed CF-netCDF WCS Encoding Profile http://www.unidata.ucar.edu/projects/THREDDS/GALEON/netCDFprofile-short.htm

- List of URLs for netCDF and CF Conventions Documents http://www.unidata.ucar.edu/projects/THREDDS/GALEON/netcdfAndCFwebpages.html
- Alternative forms for rendering netCDF metadata and data http://www.unidata.ucar.edu/projects/THREDDS/GALEON/NetCDFandStandards.htm
- ISO/TC 211, ISO/PDTS 19103 Geographic information — Conceptual Schema Language, ISO/PDTS 19103:2003.
- ISO/TC 211, ISO/FDIS 19123 Geographic information — Schema for coverage geometry and functions, ISO/FDIS 19123:2005(E).
- UNIDATA, "NetCDF", available at: http://www.unidata.ucar.edu/software/netcdf/
- BADC, "CF Convention", available at: http://badc.nerc.ac.uk/help/formats/netcdf/index_cf.html

Annex A: Details of CF-netCDF describeCoverage response (Normative)

Referring to the CF-netCDF – ISO 10123 model mapping rules, the following table maps the CF-netCDF elements into the corresponding WCS

Coverage description data structure

WCS describe Coverage element	CF-netCDF concept	Mapping Cardinality	Constraints and conditions
Domain	CoordinateSystem[a, b]	1 to 1	For each coordinate system defined in a CF-netCDF dataset a WCS coverage is generated[a, b]
Range	Dataset.Group.Variable	1 to N	All the CF-netCDF group variables defined on a same coordinate system[a, b] generate a coverage range. A group may contains variables defined on different coordinate systems[a, b].
Range	Dataset.Group	1 to 1	All the CF-netCDF group variables must be defined on the same coordinate system[a, b]
SupportedCRS	CoordinateSystem[c, a]	1 to 1	
SupportedCRS	CoordinateTransform[c, a]	1 to 1	
SupportedFormat[d]	"CF-NetCDF"		The coverage element value shall be "CF-NetCDF"

[a] For the mapping purpose, two CF-netCDF coordinate systems are considered different if they contain different spatial and temporal axes. They are not considered different if the only different axes are parametric ones (see the 11.2.5.2 paragraph).

[b] A parametric dimension may generate a Field.Axis element (see the Field data structure paragraph). Another possible strategy consists in generating a coverage Field for each

parametric axis value. That mainly depends on the parametric axis type and application.

[c] The present extension considers CF-netCDF rectified grid object (see Figure 4) and its valuation GridValueMatrix object (see 11.2.4)

[d] Naturally, it is possible to support other formats as well.

A.1 Domain of coverage data structure

WCS describe Coverage element	CF-netCDF concept	Mapping Cardinality	Constraints and conditions
SpatialDomain	CoordinateSystem.CoordinateAxis	1 to N[a]	The attribute CoordinateAxis.type must be equal to: {*Lat*, *Lon*, *Height*, *GeoX*, *GeoY*, *GeoZ*, *RadialAzimuth*, *RadialElevation*, *RadialDistance}*
TemporalDomain	CoordinateSystem.TimeAxis	1 to 1	The attribute CoordinateAxis.type must be equal to: {*time}*
[a] N = spatial axes number (i.e. 1, 2 or 3)			

A.2 Range data structure

WCS describe Coverage element	CF-netCDF concept	Mapping Cardinality	Constraints and conditions
Field	Dataset.Group.Variable	1 to 1	

A.3 Field data structure

WCS describe	CF-netCDF concept	Mapping	Constraints and conditions

Coverage element		Cardinality	
Description	Variable.name Variable.standard_name Variable.long_name Variable.Attribute[a]		
Identifier	Variable.name or Variable.standard_name	1 to 1	
Definition	Variable.units	1 to 1	
NullValue	Variable.FillValue	1 to 1	
Axis[b]	CoordinateSystem.CoordinateAxis	1 to 1	Parametric dimensions; examples are: pressure, salinity, density and CF dimensions characterized by an axis type attribute equal to: *runtime, ensemble*)[c]

[a] CF-netCDF Variable may be characterized by zero or more Attribute objects.

[b] Zero or more parametric axes are allowed

[c] It is not mandatory to generate a Field.Axis for each CF-netCDF parametric dimension. Another strategy consists in generating a coverage field for each parametric axis value. That mainly depends on the parametric axis type and application.

Annex B: Details of CF-netCDF getCoverage response (Normative)

The normal response to a valid GetCoverage operation request shall be a single coverage extracted from the coverage requested, with the specified spatial reference system, bounding box, size, format, and range subset.

The required output coverage metadata (see Table 5) or the equivalent information, shall be included with each WCS output coverage. Such equivalent metadata may be encoded within the netCDF file(s).

The GetCoverage operation response shall be XML encoded using the "Coverages" data structure specified in Annex H.2 of the OGC document OGC 07-067r5 56. This "Coverages" data structure allows the response to reference multiple files, and shall be output even when the GetCoverage operation response would otherwise contain only one file. The "Coverages" data structure is based on the Manifest data structure specified in OWS Common 1.1 [OGC 06-121r3] (see Table 4).

The encoding of the GetCoverage response consists of a "Coverages" XML document, packaged and bundled according to: the request encoding, and the value of the "store" parameter.

According to the possible "store" parameter values, two main cases are possible:

1) For "store" equal to "true", a partial GetCoverage response is transferred to the client;
2) For "store" equal to "false", a complete GetCoverage response is transferred to the client.

For a complete GetCoverage response, the server shall transfer to the client both the "Coverages" data structure and the Coverage values -e.g. through netCDF binary file(s).

For a partial GetCoverage coverage response, the server shall store the result file(s) at URL-addressable location(s) of its choosing, and return only the Coverages data structure with references to the other files.

B.1 General GetCoverage response for CF-netCDF data

A general GetCoverage response considers three sections:

ii) A "Coverages" data structure which shall reference all the data files for one output coverage, including the coverage values and any associated metadata. This is mandatory.

iii) The Required Output Coverage Metadata -or equivalent information- (see Table 5). This is mandatory.
iv) Grid Coverage(s) Values. This is optional depending on the "store" parameter value.

The abstract model of the general GetCoverage response with support for the CF-netCDF format is shown in

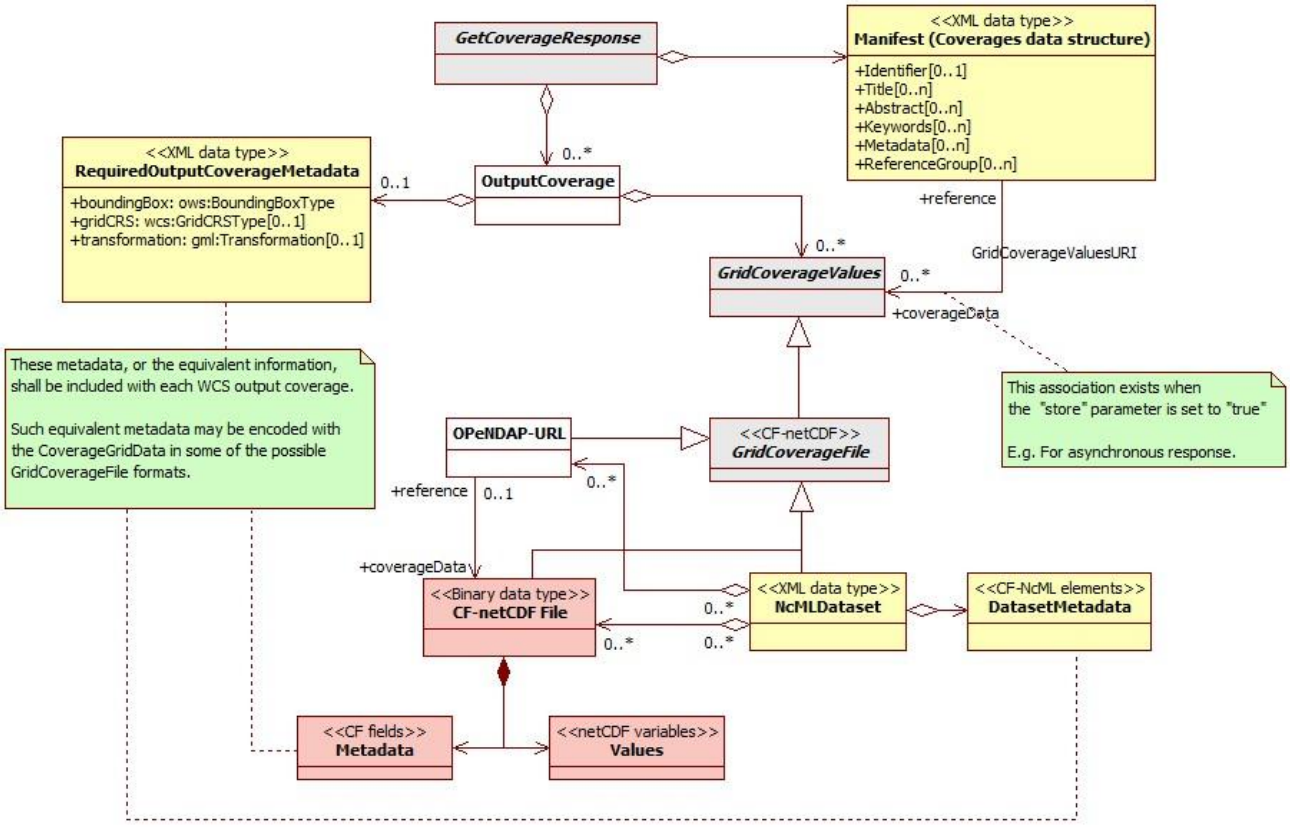

Figure 7.

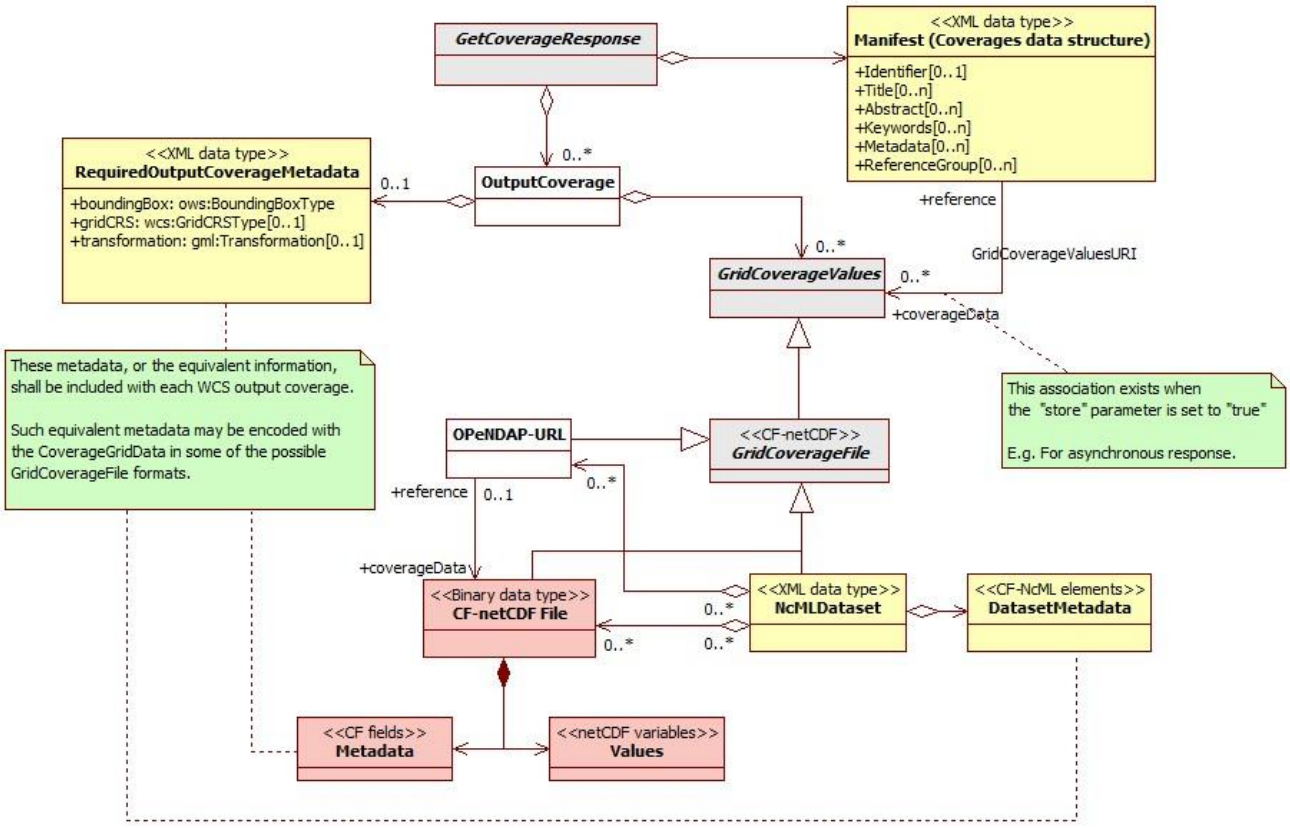

Figure 7 - Abstract model of general WCS GetCoverage response for CF-netCDF data. Yellow objects are encoded as XML documents; red objects are encoded as binary files/sections; gray objects are abstract.

Many of these classes are as per the WCS specification, but it is helpful to review them all here in the context of CF-netCDF delivery:

B.1.1 GetCoverageResponse

This is the response generated by a server to satisfy a generic getCoverage request for CF-netCDF data. The response consists of a set of OutputCoverages and a Manifest to describe the output coverages data structure.

B.1.2 OutputCoverage

This element is the coverage extracted from the coverage requested, with the specified spatial reference system, bounding box, size, format, and range subset.

It contains at least one the following objects: RequiredOutputCoverageMetadata and GridCoverageValues. When both objects are present, it is possible to have the duplication of the required output coverage metadata; in that case, the reference metadata values are the attributes of the RequiredOutputCoverageMetadata object.

When an OutputCoverage object does not contain any GridCoverageValues, then, the Manifest object must be associated with at least one GridCoverageValues object through the GridCoverageValueURI association –i.e. the "store" parameter of the GetCoverage request is set to "true".

B.1.3 GridCoverageValues

This is an abstract element referring to the values of the output coverage extracted from the coverage requested. The output coverage values are structured as grid coverage data type. This element consists of one or more GridCoverageFile, being instantiated as either a CF-netCDF3File (binary data type) or a NcMLDataset document (XML data type).

This object is optionally associated to either a CoverageOutput instance or a Manifest instance, depending on the GetCoverage "store" parameter value. In fact, the "store" parameter specifies whether response coverage should be stored, remotely from client at network URL; in that case the OutputCoverage does not include a GridCoverageValues object; while, the Manifest provides a reference to it –see the GridCoverageValuesURI association.

B.1.4 Manifest (Coverages data structure)

The manifest object is an XML document describing the content of the package containing the OutputCoverage values, returned by a GetCoverageResponse. A manifest can be used to quickly determine the content of the package without having to scan the package content (i.e. the GridCoverageValues objects content). The specified Manifest lists and describes the GridCoverageValues data structure: each resource bundled in the returned package. When the "store" parameter of the GetCoverage request is set to "true", the Manifest provides the necessary reference to the stored GridCoverageValues resources (see the GridCoverageValuesURI association). How the resources are packaged is irrelevant; for example, a package may be a zip file or a multi-part mime message [OGC 06-121r3].

According to the OGC Web Services Common Specification [OGC 06-121r3], the Manifest data structure is described in Table 4.

Table 4. Manifest data structure [OGC 06-121r3: Table 45]

Names	Definition	Data type	Multiplicity and use
identifier Identifier	An unambiguous identifier of this Manifest document, normally used by software	ows:CodeType, an adaptation of MD_Identifier class in ISO 19115 [a]	Zero or one (optional) Include when available and useful

title [e] Title	Title of this Manifest document, normally used for display to a human	LanguageString data structure, see Figure 15	Zero or more (optional)Include when available and useful [b] Include one for each language represented
abstract [e] Abstract	Brief narrative description of this Manifest document, normally available for display to a human	LanguageString data structure, see Figure 15	Zero or more (optional)Include when available and useful Include one for each language represented
keywords [e] Keywords	Unordered list of one or more commonly used or formalised word(s) or phrase(s) used to describe this Manifest document	MD_Keywords class in ISO 19115	Zero or more (optional) One for each keyword authority used
metadata Metadata	Additional metadata about this Manifest document [c]	reference to metadata or metadata contents, see gml:metaDataProperty [d]	Zero or more (optional) One for each useful metadata object
reference Group Reference Group	References to a logical group of documents or resources within this manifest document	ows:ReferenceGroup Type, see Table 46	Zero or more (optional) One for each group included

B.1.5 RequiredOutputCoverageMetadata

This object introduces the required metadata to describe each output coverage. This XML data type element is optional because the equivalent information (i.e the required output metadata) can be encoded with the coverage in the CF-netCDF file encoding. Two file encodings are allowed: netCDF binary file and, ncML document file. The metadata data structure is specified in Table 5 [OGC 07-067r5].

Table 5. *Required output coverage metadata [OGC 07-067r5]*

Names [a]	***Definition***	***Data type***	***Multiplicity and use***
boundingBox *BoundingBox*	*Bounding box that specifies extent of output coverage*	*ows:BoundingBoxType in CRS of output coverage* [b]	*One (mandatory)*
gridCRS *GridCRS*	*Definition of GridCRS used by output coverage*	*wcs:GridCRSType*	*Zero or one (conditional)* *Include when GridCRS used by BoundingBox*
Transformation *Transformation*	*Definition of georeferencing coordinate transformation*	*gml:Transformation or gml:ConcatenatedOperation*	*Zero or one (conditional)* *Include when output coverage is not georectified but is georeferenced*
a See Table 1 of [OGC 06-121r3] for UML and XML naming conventions. *b The Coordinate Reference System (CRS) of an output coverage may be either a GridCRS or an ImageCRS.*			

B.1.6 GridCoverageFile

This abstract element specializes a GridCoverageValue being instantiated as a CF-netCDF coverage dataset; the following encoding possibilities are foreseen: a) a CF-netCDF3File (binary data type); b) a NcMLDataset document (XML data type); c) an OPeNDAP reference. It contains the values, and optionally the required metadata, of an Output Coverage.

This element may be either bundled in the package containing the OutputCoverage values (i.e. “store” parameter = “false”), or stored at an URI-addressable location (i.e. “store” parameter = “true”).

B.1.7 GridCoverageValuesURI association

When the GetCoverage “store” parameter is set to “true”, OutputCoverage does not contain any GridCoverageValues object. In fact, the GridCoverageFile element(s) containing the coverage values, and optionally the required metadata, are stored at an URI-addressable location. These resources are referenced by the Manifesto through this association.

B.1.8 CF-netCDF file

This element realizes a GridCoverageFile. It represents a netCDF ver. 3.0 binary file, complying with the CF ver. 1.1 conventions. The element contains data Values, encoded as netCDF variables, and Metadata encoded as CF fields. These metadata can contain the required information for each output coverage, avoiding to include the RequiredOutputCoverageMetadata element.

A CF-netCDF file may be accessed directly or can be referenced by: a) a ncML dataset; b) an OPeNDAP URL.

B.1.9 NcMLDataset

This element realizes a GridCoverageFile. It represents a ncML ver. 1.0 document. A ncML document provides the XML encoding of one or more CF-netCDF file. For performances sake, a ncML document generally encodes only the CF-netCDF metadata leaving the data values in the binary format and providing pointers (i.e. references) to those. Thus, the element contains a mandatory MetadataDataset object (i.e. the encoded file metadata), and optionally points to the binary file(s) for values. The MetadataDataset can contain the required information for each output coverage, avoiding to include the RequiredOutputCoverageMetadata element.

A ncML document may point to the data values: a) directly –i.e. referencing a CF-netCDF file; b) including an OPeNDAP URL –which in turn reference a CF-netCDF file.

B.1.10 OPeNDAP-URL

This element realizes a GridCoverageFile. Actually, this is a reference to a CF-netCDF file. In fact, OPeNDAP is a relatively simple approach (Web protocol) that allows applications to access remote, time-aggregated collections of netCDF-CF files (virtual datasets –often terabyte sized) through the unaltered netCDF API –as if they were local netCDF files.

An OPeNDAP-URL instance may also be contained by a NcMLDataset object to reference a CF-netCDF file –the actual binary data.

B.2 Content model of the WCS complete GetCoverage response for CF-netCDF3 binary file

This section describes the package returned by a complete GetCoverage response packaging CF-netCDF3 binary file(s).

Figure 8 shows the corresponding bundle structure transferred by a WCS Complete GetCoverage response for CF-netCDF3 binary file.

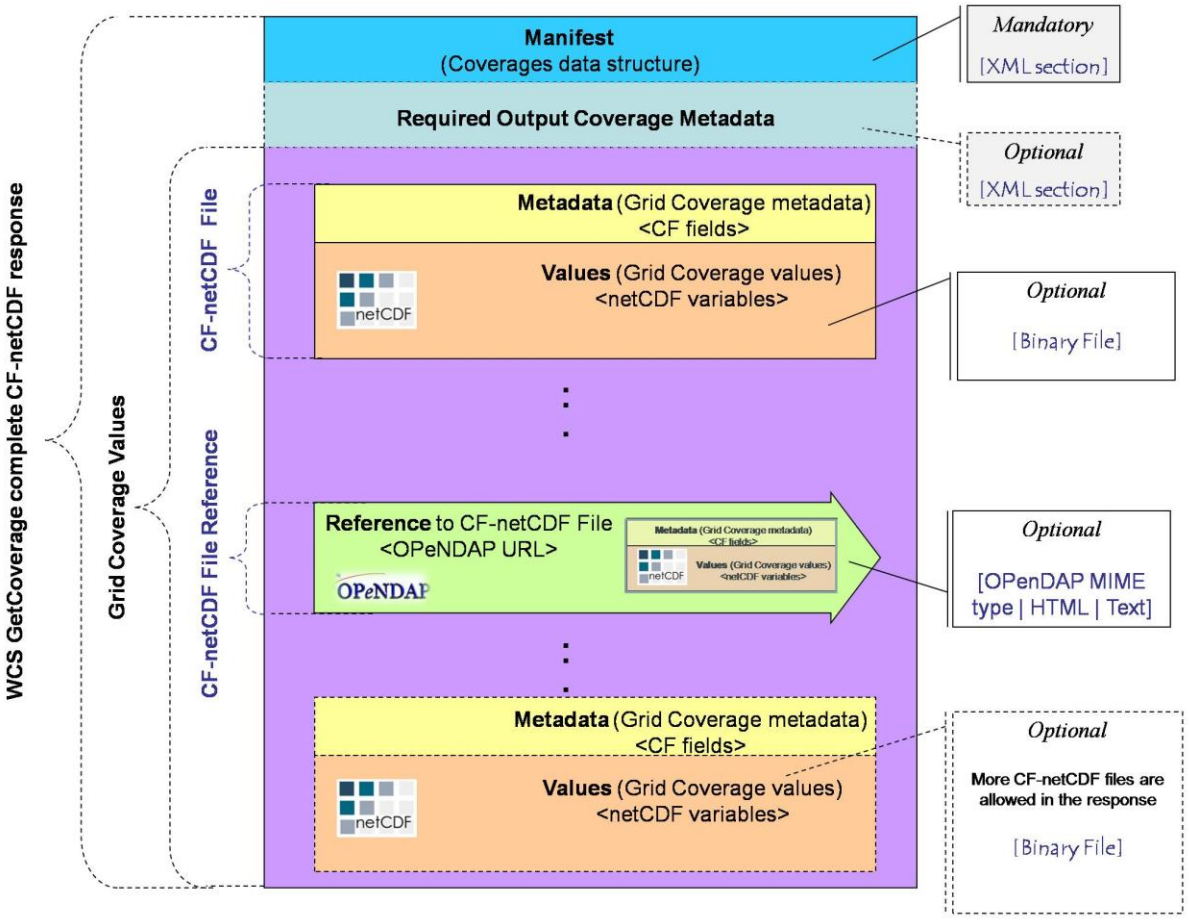

Figure 8 - Data bundle structure transferred by a general WCS complete GetCoverage response returning a CF-netCDF3 binary file

The Required Output Coverage Metadata XML section is optional. In fact, the required metadata can be encoded in the CF-netCDF data file.

If present, the Required Output Coverage Metadata XML section should use the XML encoding referenced in Sub-clause 10.3.9 of the OGC 07-067r5 document.

If the Required Output Coverage Metadata XML section is not present, then the CF-netCDF grid coverage file(s) must contain the equivalent information. The structure of these metadata elements is described in the A.1.1 (Implementing the required output coverage metadata as CF fields) paragraph.

If metadata is present in both Required Output Coverage Metadata section and in the CF-netCDF files, then duplicate fields should be consistent. (Note: This situation is strongly discouraged; nevertheless, it may be useful to encode the CF elements in XML in order to support rapid service chaining applications). In that case the Required Output Coverage Metadata section is to be considered as the reference one.

The GridCoverageValue section of the response may contain: a) CF-netCDF3 binary file(s); b) OPeNDAP URL(s) which point(s) to CF-netCDF3 binary file(s).

OPenDAP URL may be encoded using one of the following options: i) a specific MIME type for an OPeNDAP endpoint –analogous to application/x-jpip-xml; ii) an HTML document containing the OPeNDAP URL; iii) a simple text –semantically, this is the worst solution.

Since both XML Manifest and Grid Coverage Values (i.e. netCDF3 binary file(s) and/or OPeNDAP URLs) are mandatory, then a WCS Complete GetCoverage response must be a multipart message.

B.3 Complete GetCoverage response for ncML document

The netCDF community makes use of ncML to encode complex netCDF data structures. In fact it is possible to use ncML elements to define a virtual netCDF dataset which consists of one or more netCDF data files.

In a complete GetCoverage response returning a ncML/CF-netCDF document, the Grid CoverageValues are encoded and returned according to one or more of the following options:

1. XML encoded values conforming to the ncML schema: this ncML section encodes the content of a CF-netCDF binary file.
2. NetCDF3 binary files (i.e. *GridCoverageFile* instances) attached to an ncML document.
3. OPenDAP URL encoded in the ncML schema: actually, this ncML section references the content of a CF-netCDF3 binary file.

In any case, the coverage metadata section is encoded in the ncML document as elements conforming to the ncML schema.

Therefore, it is possible to find the same metadata content both in the Coverage metadata section and in the attached netCDF3 binary file(s), expressed as CF fields. In that case, to avoid an inconsistency situation, the ncML document elements must be considered as the reference metadata values.

This type on encoding allows to return very complex packages; for example, more than one output coverage (i.e. dataset) each of them containing/referring-to more than one CF-netCDF file.

Figure 9 the corresponding bundle structure transferred by a WCS Complete GetCoverage response for ncML documents

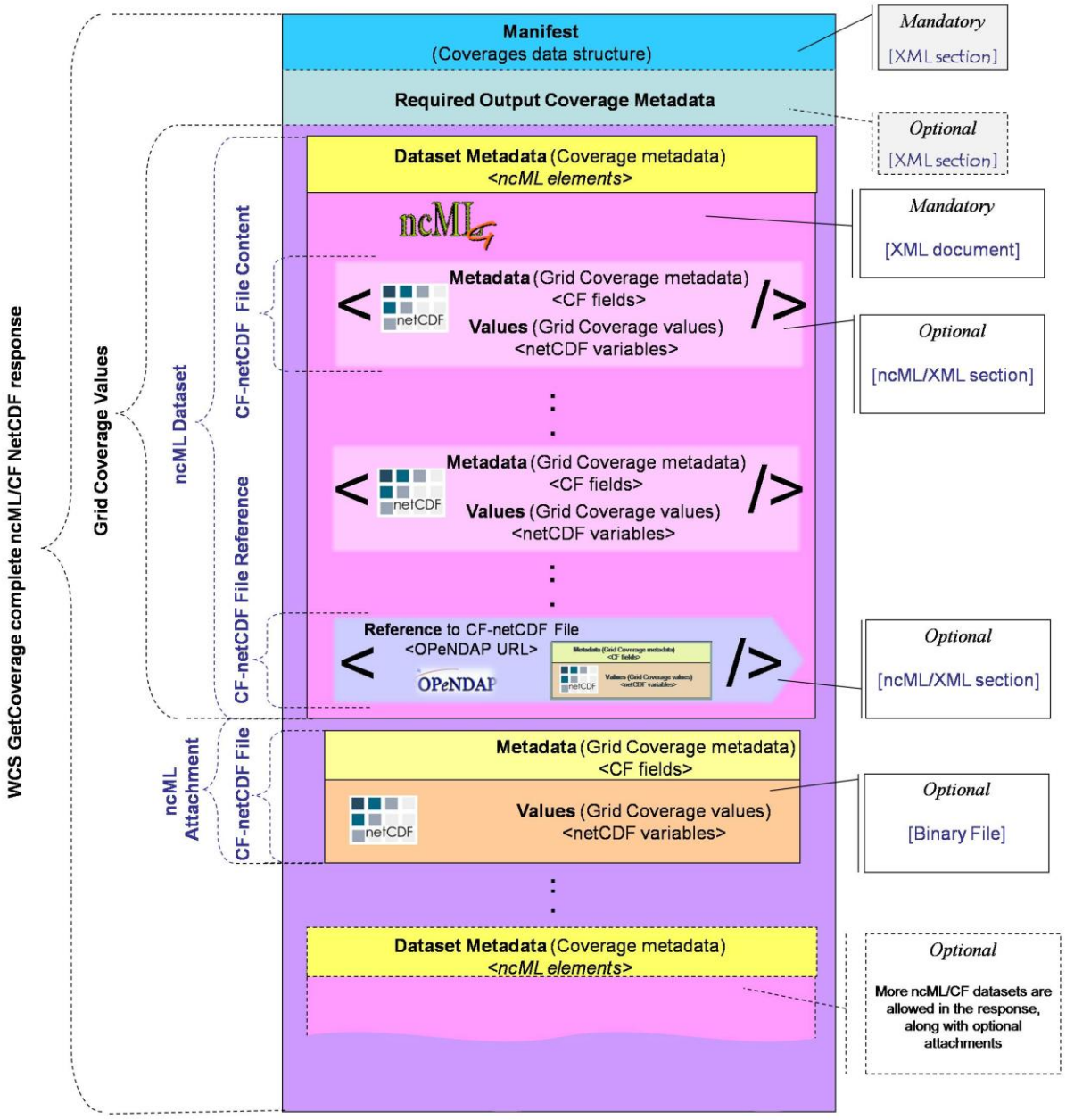

Figure 9 - Data bundle structure transferred by a general WCS complete GetCoverage response returning a ncML/CF-netCDF document

It is noteworthy that there is a new data structure level: the ncML dataset which is described by its own metadata (i.e. ncML elements) encoded in XML; a dataset encodes/refers to one or more CF-netCDF datafiles; one or more these files may be

enclosed as attachments to the ncML document. Others may be directly encoded in the ncML dataset document. Eventually, others may be referred through OPenDAP URLs.

B.5 Partial GetCoverage response

A partial coverage response refers to a coverage request characterized by the "store" output attribute set to "true".

If the “store” parameter has the value “true”, the server shall store the result file(s) at URL-addressable location(s) of its choosing, and return only the Coverages data structure with references to the other files.

This output structure element specifies whether response coverage should be stored remotely from client at a network URL (more generally a URI), instead of being returned within the operation response. The remote URI is contained in the getCoverage response.

The default value is "false". A partial response is useful to enable asynchronous requests to a WCS server which is able to support this type of functionality.

The partial getCoverage response may consist of the “Coverages” data structure (the *ows:coverage* element) containing the required output metadata (i.e. the *ows:CoverageMetadata)*; the remote URI is encoded in this section.

Another option consists of returning a ncML based response. This XML document contains the required coverage output metadata and the the remote URI, where to retrieve the netCDF file(s).

B.6 WCS GetCoverage response: Multipart data encoding

Multipart data encoding is a technique for transferring multiple contents inside a single HTTP Response. In the context of WCS GetCoverage responses it can be adopted whenever a binary content must be transferred.

Different strategies should be adopted for the different possible cases. Table 6 shows the four possibilities:

Table 6. *Multipart possible strategies*

Case	Request	Target	Possible Responses	Specification
#1	SOAP	binary data	SOAP with binary	SOAP Messages with Attachments

			Attachment (e.g. manifest and binary coverages content)	1.2
#2	HTTP (GET or POST)	binary data	MIME Multipart with binary parts	MIME multipart/related
#3	SOAP	ncML data	SOAP with ncML Attachment (e.g. manifest and ncML coverages)	SOAP Messages with Attachments 1.2 / MTOM-XOP
#4	HTTP (GET or POST)	ncML data	MIME Multipart with ncML parts	MIME multipart/related

The WCS GetCoverage response can be encoded in a Multipart message. The encoding strategy is different depending on the use of SOAP or HTTP binding.

B.6.1 Case #1 SOAP with binary data

In case of SOAP with binary data, the response can refer to local or remote resources using the Manifest. Local resources are always attached to the message, while remote resources can be attached as a copy to avoid multiple access. The SOAP response must be encoded along with its attachments in a MIME multipart/related message using the following rules:

1) A world-unique content MUST be assigned to each attached binary content.

2) In the Manifest, every reference to the local binary contents MUST be transformed in a related-part reference according to the "cid" schema [RFC 2387] using the proper Content-ID. References to the attached remote binary contents MAY be transformed in a related-part reference according to the "cid" schema [RFC 2387] using the proper Content-ID;

3) The first part of the Multipart message (root) MUST contain the SOAP Envelope including the Manifest.

4) Each one of the following parts MUST include one binary content. The Content-ID header refers to the corresponding content identifier. For remote resources, the Content-Location header refers to the real location of the binary content.

Example (SOAP Request of two netCDF data items and metadata)

```
MIME-Version: 1.0
```

```
Content-Type: multipart/related; boundary="----=_NextPart_000_0000_93251752.3C5526C0"


------=_NextPart_000_0000_93251752.3C5526C0

Content-Type: application/xml; charset="UTF-8"


<?xml version="1.0" encoding="UTF-8"?>

<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope">

  <soap:Header/>

  <soap:Body>

    <Coverages xmlns="http://www.opengis.net/wcs/1.1/ows" xmlns:ows="http://www.opengis.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:schemaLocation="http://www.opengis.net/wcs/1.1/ows ../owsCoverages.xsd
http://www.opengis.net/ows ../../../ows/1.0.0/ows19115subset.xsd">

      <Coverage>

        <ows:Title>Example</ows:Title>

        <ows:Abstract>Example of coverage</ows:Abstract>

        <Identifier>Example</Identifier>

        <Reference href="cid:bfa8e8ac@example.net" />

        <Reference href="cid:c3499c27@example.net" />

        <Reference href="http://example.net/data/data2.nc" />

      </Coverage>

    </Coverages>

  </soap:Body>

</soap:Envelope>

------=_NextPart_000_0000_93251752.3C5526C0

Content-Type: application/xml

Content-ID: <bfa8e8ac@example.net>


...(metadata content)...

------=_NextPart_000_0000_93251752.3C5526C0

Content-Type: application/CF-netCDF3
```

```
Content-Transfer-Encoding: base64

Content-ID: <c3499c27@example.net>



...(1st data content in netCDF)...

------=_NextPart_000_0000_93251752.3C5526C0

Content-Type: application/CF-netCDF3

Content-Transfer-Encoding: base64

Content-ID: <7904e041@example.net>

Content-Location: http://example.net/data/data2.nc



...(2nd data content in netCDF)...

------=_NextPart_000_0000_93251752.3C5526C0--
```

B.6.2 Case #2 HTTP Response with binary data

In case of a HTTP Response with binary data, the response can refer to local or remote resources using the Manifest. Local resources are always attached to the message, while remote resources can be attached as a copy to avoid multiple access. The HTTP response must be encoded along with its attachments in a MIME multipart/related message using the following rules:

1) A world-unique content MUST be assigned to each attached binary content.

2) In the Manifest, every reference to the local binary contents MUST be transformed in a related-part reference according to the "cid" schema [RFC 2387] using the proper Content-ID. References to the attached remote binary contents MAY be transformed in a related-part reference according to the "cid" schema [RFC 2387] using the proper Content-ID;

3) The first part of the Multipart message (root) MUST contain the SOAP Envelope including the Manifest..

4) Each one of the following parts MUST include one binary content. The Content-ID header refers to the corresponding content identifier. For remote resources, the Content-Location header refers to the real location of the binary content.

Example (HTTP Request of two netCDF data items and metadata)

```
MIME-Version: 1.0
```

```
Content-Type: multipart/related; boundary="----=_NextPart_000_0000_93251752.3C5526C0"


------=_NextPart_000_0000_93251752.3C5526C0

Content-Type: application/xml; charset="UTF-8"


<?xml version="1.0" encoding="UTF-8"?>

<Coverages xmlns="http://www.opengis.net/wcs/1.1/ows" xmlns:ows="http://www.opengis.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:schemaLocation="http://www.opengis.net/wcs/1.1/ows ../owsCoverages.xsd
http://www.opengis.net/ows ../../../ows/1.0.0/ows19115subset.xsd">

  <Coverage>

    <ows:Title>Example</ows:Title>

    <ows:Abstract>Example of coverage</ows:Abstract>

    <Identifier>Example</Identifier>

    <Reference href="cid:bfa8e8ac@example.net" />

    <Reference href="cid:c3499c27@example.net" />

    <Reference href="http://example.net/data/data2.nc" />

  </Coverage>

</Coverages>

------=_NextPart_000_0000_93251752.3C5526C0

Content-Type: application/xml

Content-ID: <bfa8e8ac@example.net>


...(metadata content)...

------=_NextPart_000_0000_93251752.3C5526C0

Content-Type: application/CF-netCDF3

Content-Transfer-Encoding: base64

Content-ID: <c3499c27@example.net>


...(1st data content in netCDF)...

------=_NextPart_000_0000_93251752.3C5526C0
```

```
Content-Type: application/CF-netCDF3

Content-Transfer-Encoding: base64

Content-ID: <7904e041@example.net>

Content-Location: http://example.net/data/data2.nc


...(2nd data content in netCDF)...

------=_NextPart_000_0000_93251752.3C5526C0--
```

B.6.3 Case #3 and #4: SOAP or HTTP Response with NcML data

In case of ncML data, the response can be encoded as in cases #1 and #2. The only significant differences is that one or more multipart sections will contain XML data instead of binary data.

Example (SOAP Response with binary and ncML data)

```
MIME-Version: 1.0

Content-Type: multipart/related; boundary="----=_NextPart_000_0000_93251752.3C5526C0"


------=_NextPart_000_0000_93251752.3C5526C0

Content-Type: application/xml; charset="UTF-8"


<?xml version="1.0" encoding="UTF-8"?>

<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope">

  <soap:Header/>

  <soap:Body>

    <Coverages xmlns="http://www.opengis.net/wcs/1.1/ows" xmlns:ows="http://www.opengis.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:schemaLocation="http://www.opengis.net/wcs/1.1/ows ../owsCoverages.xsd
http://www.opengis.net/ows ../../../ows/1.0.0/ows19115subset.xsd">

      <Coverage>

        <ows:Title>Example</ows:Title>

        <ows:Abstract>Example of coverage</ows:Abstract>
```

```
        <Identifier>Example</Identifier>
        <Reference href="cid:bfa8e8ac@example.net" />
        <Reference href="cid:c3499c27@example.net" />
        <Reference href="http://example.net/data/data2.ncml" />
      </Coverage>
    </Coverages>
  </soap:Body>
</soap:Envelope>
------=_NextPart_000_0000_93251752.3C5526C0
Content-Type: application/xml
Content-ID: <bfa8e8ac@example.net>

...(metadata content)...
------=_NextPart_000_0000_93251752.3C5526C0
Content-Type: application/CF-netCDF3
Content-Transfer-Encoding: base64
Content-ID: <c3499c27@example.net>

...(data content in netCDF)...
------=_NextPart_000_0000_93251752.3C5526C0
Content-Type: application/CF-netCDF3
Content-Transfer-Encoding: base64
Content-ID: <1324bc12@example.net>
Content-Location: http://example.net/data/data2.ncml

...(data content in ncML)...
------=_NextPart_000_0000_93251752.3C5526C0--
```

The reason why it is useful to distinguish between the ncML case and binary data cases is that ncML documents can contain binary sections in base64 encoding. They can be left enclosed in the ncML document or, for efficiency purposes, they can be extracted and

encoded in a format other than base64. Such improvement can be obtained serializing the ncML document using nested multipart messages according to the MTOM/XOP specifications.

The following two encoding examples reports multipart sections for ncML with binary data included and extracted using XOP, respectively.

B.6.4.1 Binary data included

Example

multipart section containing ncML with binary data included (pale blue background)

```
------=_NextPart_000_0000_93251752.3C5526C0
Content-Type: application/CF-netCDF3
Content-Transfer-Encoding: base64
Content-ID: <1324bc12@example.net>
Content-Location: http://example.net/data/data2.ncml

<?xml version="1.0" encoding="UTF-8"?>
<netcdf xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://www.unidata.ucar.edu/namespaces/netcdf/ncml-2.2">
   <dimension name="longitude" length="96"/>
   <dimension name="bounds_axis" length="2"/>
   <dimension name="latitude" length="73"/>
   <dimension name="air_pressure" length="15"/>
   <dimension name="time" length="140"/>
   <attribute name="Conventions" type="string" value="CF-1.1"/>
   <attribute name="source" type="string" value="Data from model run ABC123"/>
   <variable name="longitude" shape="longitude" type="double">
      <attribute name="standard_name" type="string" value="longitude"/>
      <attribute name="units" type="string" value="degrees_east"/>
      <attribute name="bounds" type="string" value="bound_longitude"/>
      <attribute name="axis" type="string" value="X"/>
      <values increment="3.75" npts="96" start="0.0"/>
```

```
</variable>
<variable name="bound_longitude" shape="longitude bounds_axis" type="double"> </variable>
<variable name="latitude" shape="latitude" type="double">
   <attribute name="standard_name" type="string" value="latitude"/>
   <attribute name="units" type="string" value="degrees_north"/>
   <attribute name="bounds" type="string" value="bound_latitude"/>
   <attribute name="axis" type="string" value="Y"/>
   <values increment="-2.5" npts="73" start="90.0"/>
</variable>
<variable name="bound_latitude" shape="latitude bounds_axis" type="double"> </variable>
<variable name="air_pressure" shape="air_pressure" type="float">
   <attribute name="standard_name" type="string" value="air_pressure"/>
   <attribute name="units" type="string" value="hPa"/>
   <attribute name="axis" type="string" value="Z"/>
   <attribute name="positive" type="string" value="down"/>
   <values increment="70.71429" npts="15" start="10.0"/>
</variable>
<variable name="time" shape="time" type="double">
   <attribute name="calendar" type="string" value="360_day"/>
   <attribute name="standard_name" type="string" value="time"/>
   <attribute name="units" type="string" value="days since 2289-1-1"/>
   <attribute name="bounds" type="string" value="bound_time"/>
   <attribute name="axis" type="string" value="T"/>
   <values increment="360.0" npts="140" start="510.0"/>
</variable>
<variable name="bound_time" shape="time bounds_axis" type="double"> </variable>
<variable name="air_temperature" shape="time air_pressure latitude longitude" type="float">
   <attribute name="standard_name" type="string" value="air_temperature"/>
   <attribute name="units" type="string" value="Kir_temperature"/>
   <attribute name="long_name" type="string" value="temperature on pressure level"/>
```

```
      <attribute name="cell_methods" type="string" value="longitude: latitude: mean time: mean (interval: 4 h)"/>

      <attribute name="_FillValue" type="float" value="-1.073742e+09"/>

      <values xm:xmlmime="binary/octet-stream">Li4uKGJhc2U2NCBlbmNvZGVkIGRhdGEpLi4u...(base64 encoded data)...BlbmNvZGVkIGRhdGEpLi4u==</values>

   </variable>

</netcdf>

------=_NextPart_000_0000_93251752.3C5526C0--
```

B.6.4.2 Binary extracted using XOP

Example

multipart section containing ncML with binary data extracted using XOP (pale blue background)

```
------=_NextPart_000_0000_93251752.3C5526C0

MIME-Version: 1.0

Content-Type: Multipart/Related;boundary=MIME_boundary;  type="application/xop+xml";

   start="<12dea45c@example.net >";

   startinfo="application/ncML+xml;


--MIME_boundary

Content-Type: application/xop+xml; charset=UTF-8;

   type="application/ncML+xml; action=\"ProcessData\""

Content-ID: <12dea45c@example.net>

Content-Location: http://example.net/data/data2.ncml


<?xml version="1.0" encoding="UTF-8"?>

<netcdf xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

   xmlns="http://www.unidata.ucar.edu/namespaces/netcdf/ncml-2.2">

   <dimension name="longitude" length="96"/>

   <dimension name="bounds_axis" length="2"/>

   <dimension name="latitude" length="73"/>
```

```
<dimension name="air_pressure" length="15"/>
<dimension name="time" length="140"/>
<attribute name="Conventions" type="string" value="CF-1.1"/>
<attribute name="source" type="string" value="Data from model run ABC123"/>
<variable name="longitude" shape="longitude" type="double">
   <attribute name="standard_name" type="string" value="longitude"/>
   <attribute name="units" type="string" value="degrees_east"/>
   <attribute name="bounds" type="string" value="bound_longitude"/>
   <attribute name="axis" type="string" value="X"/>
   <values increment="3.75" npts="96" start="0.0"/>
</variable>
<variable name="bound_longitude" shape="longitude bounds_axis" type="double"> </variable>
<variable name="latitude" shape="latitude" type="double">
   <attribute name="standard_name" type="string" value="latitude"/>
   <attribute name="units" type="string" value="degrees_north"/>
   <attribute name="bounds" type="string" value="bound_latitude"/>
   <attribute name="axis" type="string" value="Y"/>
   <values increment="-2.5" npts="73" start="90.0"/>
</variable>
<variable name="bound_latitude" shape="latitude bounds_axis" type="double"> </variable>
<variable name="air_pressure" shape="air_pressure" type="float">
   <attribute name="standard_name" type="string" value="air_pressure"/>
   <attribute name="units" type="string" value="hPa"/>
   <attribute name="axis" type="string" value="Z"/>
   <attribute name="positive" type="string" value="down"/>
   <values increment="70.71429" npts="15" start="10.0"/>
</variable>
<variable name="time" shape="time" type="double">
   <attribute name="calendar" type="string" value="360_day"/>
   <attribute name="standard_name" type="string" value="time"/>
```

```
        <attribute name="units" type="string" value="days since 2289-1-1"/>
        <attribute name="bounds" type="string" value="bound_time"/>
        <attribute name="axis" type="string" value="T"/>
        <values increment="360.0" npts="140" start="510.0"/>
    </variable>
    <variable name="bound_time" shape="time bounds_axis" type="double"> </variable>
    <variable name="air_temperature" shape="time air_pressure latitude longitude" type="float">
        <attribute name="standard_name" type="string" value="air_temperature"/>
        <attribute name="units" type="string" value="Kir_temperature"/>
        <attribute name="long_name" type="string" value="temperature on pressure level"/>
        <attribute name="cell_methods" type="string" value="longitude: latitude: mean time: mean (interval:
4 h)"/>
        <attribute name="_FillValue" type="float" value="-1.073742e+09"/>
        <values xm:xmlmime="binary/octet-stream"><xop:Include
    xmlns:xop='http://www.w3.org/2004/08/xop/include'
    href='cid:bd43gh2y@example.net'/>
        </values>
    </variable>
</netcdf>
--MIME_boundary
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary
Content-ID: <bd43gh2y@example.net>


...(binary encoded data)...
--MIME_boundary--
------=_NextPart_000_0000_93251752.3C5526C0--
```

B.6.5 Content-ID generation

According to [RFC 2045] "Content-ID values must be generated to be world-unique". In WCS responses it is suggested that Content-ID has the following format:

Content-ID : = Local “@” Domain

where Local is a locally unique identifier and Domain is an Internet domain administered by the coverages provider. The locally unique identifier is obscure. This means that no semantic is required to be associated to the local name. For example it could be generated with a hashing function from coverages metadata (such as in the examples above).

B.6.6 Proposed extensions for handling ncML Responses

When the GetCoverage Response includes only references to one or more ncML data items, the method described in Case 3 above can be adopted. Anyway it could pose problems related to the multipart/related encoding.

B.6.6.1 Problem

The GetCoverage Response contains nested multipart/related messages. Not all of multipart parsers are capable of handling nested multipart messages.

B.6.6.2 Proposed Solution

To extend the Manifest specifications to allow also insertion of Data Items besides references.

B.6.6.3 Motivation

The problem above is due to the fact that the Manifest specifications require that Data Items are externally referred using a ReferenceGroup (such as Coverage). This requires that external resources are attached to the Manifest as multipart sections. If a section is a XML document again containing binary sections, another multipart encoding is required generating nested multipart messages which are syntactically and semantically correct, but difficult to handle.

If the Manifest could include Data Items instead of reference to them, the encoding could be made much easier using MTOM/XOP generating plain (not nested) multipart messages. This requires that a ReferenceGroup could be also a XML fragment (e.g. the ncML root element). It is noteworthy that the Data Item should be logically included in the Manifest, but it could be physically located externally, for example by use of XInclude specifications.

Therefore, a good solution to possible technological problem in using nested multipart messages, seems to be the inclusion of the ncML root element as a Manifest Data Item (e.g. directly or using XInclude). In fact, ncML (that is the netCDF XML encoding) seems to be particularly suitable for being dispatched with MTOM/XOP.

From this perspective the undergoing work to introduce a full complaint GML application profile for netCDF-CF (see the ncML-G+ specification based on the ncML-Gml) is promising.

Appendix C: Examples

C.1 Example: netCDF 3 with CF1.1 convention dataset

```
netcdf air_temperature {
dimensions:
        longitude = 96 ;
        bounds_axis = 2 ;
        latitude = 73 ;
        air_pressure = 15 ;
        time = 140 ;
variables:
        double longitude(longitude) ;
                longitude:standard_name = "longitude" ;
                longitude:units = "degrees_east" ;
                longitude:bounds = "bound_longitude" ;
                longitude:axis = "X" ;
                double bound_longitude(longitude, bounds_axis) ;
        double latitude(latitude) ;
                latitude:standard_name = "latitude" ;
                latitude:units = "degrees_north" ;
                latitude:bounds = "bound_latitude" ;
                latitude:axis = "Y" ;
                double bound_latitude(latitude, bounds_axis) ;
        float air_pressure(air_pressure) ;
                air_pressure:standard_name = "air_pressure" ;
                air_pressure:units = "hPa" ;
                air_pressure:axis = "Z" ;
                air_pressure:positive = "down" ;
        double time(time) ;
                time:calendar = "360_day" ;
```

```
            time:standard_name = "time" ;
            time:units = "days since 2289-1-1" ;
            time:bounds = "bound_time" ;
            time:axis = "T" ;
            double bound_time(time, bounds_axis) ;
      float air_temperature(time, air_pressure, latitude, longitude) ;
            air_temperature:standard_name = "air_temperature" ;
            air_temperature:units = "K" ;
            air_temperature:long_name = "temperature on pressure levels" ;
            air_temperature:cell_methods = "longitude: latitude: mean time: mean (interval: 4 h)" ;
            air_temperature:_FillValue = -1.073742e+09f ;

// global attributes:
            :Conventions = "CF-1.1" ;
            :source = "Data from model run ABC123" ;
data:

 longitude = 0, 3.75, 7.5, 11.25, 15, 18.75, 22.5, 26.25, 30, 33.75, 37.5,
   41.25, 45, 48.75, 52.5, 56.25, 60, 63.75, 67.5, 71.25, 75, 78.75, 82.5,
   86.25, 90, 93.75, 97.5, 101.25, 105, 108.75, 112.5, 116.25, 120, 123.75,
   127.5, 131.25, 135, 138.75, 142.5, 146.25, 150, 153.75, 157.5, 161.25,
   165, 168.75, 172.5, 176.25, 180, 183.75, 187.5, 191.25, 195, 198.75,
   202.5, 206.25, 210, 213.75, 217.5, 221.25, 225, 228.75, 232.5, 236.25,
   240, 243.75, 247.5, 251.25, 255, 258.75, 262.5, 266.25, 270, 273.75,
   277.5, 281.25, 285, 288.75, 292.5, 296.25, 300, 303.75, 307.5, 311.25,
   315, 318.75, 322.5, 326.25, 330, 333.75, 337.5, 341.25, 345, 348.75,
   352.5, 356.25 ;
```

C.2 Example for a ncML dataset

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<netcdf xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://www.unidata.ucar.edu/namespaces/netcdf/ncml-2.2">
   <dimension name="longitude" length="96"/>
   <dimension name="bounds_axis" length="2"/>
   <dimension name="latitude" length="73"/>
   <dimension name="air_pressure" length="15"/>
   <dimension name="time" length="140"/>
   <attribute name="Conventions" type="string" value="CF-1.1"/>
   <attribute name="source" type="string" value="Data from model run ABC123"/>
   <variable name="longitude" shape="longitude" type="double">
      <attribute name="standard_name" type="string" value="longitude"/>
      <attribute name="units" type="string" value="degrees_east"/>
      <attribute name="bounds" type="string" value="bound_longitude"/>
      <attribute name="axis" type="string" value="X"/>
      <values increment="3.75" npts="96" start="0.0"/>
   </variable>
   <variable name="bound_longitude" shape="longitude bounds_axis" type="double"> </variable>
   <variable name="latitude" shape="latitude" type="double">
      <attribute name="standard_name" type="string" value="latitude"/>
      <attribute name="units" type="string" value="degrees_north"/>
      <attribute name="bounds" type="string" value="bound_latitude"/>
      <attribute name="axis" type="string" value="Y"/>
      <values increment="-2.5" npts="73" start="90.0"/>
   </variable>
   <variable name="bound_latitude" shape="latitude bounds_axis" type="double"> </variable>
   <variable name="air_pressure" shape="air_pressure" type="float">
      <attribute name="standard_name" type="string" value="air_pressure"/>
      <attribute name="units" type="string" value="hPa"/>
      <attribute name="axis" type="string" value="Z"/>
      <attribute name="positive" type="string" value="down"/>
      <values increment="70.71429" npts="15" start="10.0"/>
   </variable>
   <variable name="time" shape="time" type="double">
      <attribute name="calendar" type="string" value="360_day"/>
      <attribute name="standard_name" type="string" value="time"/>
      <attribute name="units" type="string" value="days since 2289-1-1"/>
      <attribute name="bounds" type="string" value="bound_time"/>
      <attribute name="axis" type="string" value="T"/>
      <values increment="360.0" npts="140" start="510.0"/>
   </variable>
   <variable name="bound_time" shape="time bounds_axis" type="double"> </variable>
   <variable name="air_temperature" shape="time air_pressure latitude longitude" type="float">
      <attribute name="standard_name" type="string" value="air_temperature"/>
      <attribute name="units" type="string" value="Kir_temperature"/>
      <attribute name="long_name" type="string" value="temperature on pressure level"/>
      <attribute name="cell_methods" type="string" value="longitude: latitude: mean time: mean (interval:
4 h)"/>
      <attribute name="_FillValue" type="float" value="-1.073742e+09"/>
   </variable>
</netcdf>
```

C.3 GetCoverage response encoding examples

Detailed examples of GetCoverage response encodings based on SOAP and Multipart-related technology. -

C.3.1 SOAP Request of two netCDF data items and metadata

```
MIME-Version: 1.0

Content-Type: multipart/related; boundary="----=_NextPart_000_0000_93251752.3C5526C0"


------=_NextPart_000_0000_93251752.3C5526C0

Content-Type: application/xml; charset="UTF-8"


<?xml version="1.0" encoding="UTF-8"?>

<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope">

  <soap:Header/>

  <soap:Body>

    <Coverages xmlns="http://www.opengis.net/wcs/1.1/ows" xmlns:ows="http://www.opengis.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:schemaLocation="http://www.opengis.net/wcs/1.1/ows ../owsCoverages.xsd
http://www.opengis.net/ows ../../../ows/1.0.0/ows19115subset.xsd">

      <Coverage>

        <ows:Title>Example</ows:Title>

        <ows:Abstract>Example of coverage</ows:Abstract>

        <Identifier>Example</Identifier>

        <Reference href="cid:bfa8e8ac@example.net" />

        <Reference href="cid:c3499c27@example.net" />

        <Reference href="http://example.net/data/data2.nc" />

      </Coverage>

    </Coverages>

  </soap:Body>

</soap:Envelope>

------=_NextPart_000_0000_93251752.3C5526C0
```

```
Content-Type: application/xml

Content-ID: <bfa8e8ac@example.net>


...(metadata content)...

------=_NextPart_000_0000_93251752.3C5526C0

Content-Type: application/CF-netCDF3

Content-Transfer-Encoding: base64

Content-ID: <c3499c27@example.net>


...(1st data content in netCDF)...

------=_NextPart_000_0000_93251752.3C5526C0

Content-Type: application/CF-netCDF3

Content-Transfer-Encoding: base64

Content-ID: <7904e041@example.net>

Content-Location: http://example.net/data/data2.nc


...(2nd data content in netCDF)...

------=_NextPart_000_0000_93251752.3C5526C0--
```

C.3.2 HTTP Request of two netCDF data items and metadata

```
MIME-Version: 1.0

Content-Type: multipart/related; boundary="----=_NextPart_000_0000_93251752.3C5526C0"


------=_NextPart_000_0000_93251752.3C5526C0

Content-Type: application/xml; charset="UTF-8"


<?xml version="1.0" encoding="UTF-8"?>

<Coverages xmlns="http://www.opengis.net/wcs/1.1/ows" xmlns:ows="http://www.opengis.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:schemaLocation="http://www.opengis.net/wcs/1.1/ows ../owsCoverages.xsd
http://www.opengis.net/ows ../../../ows/1.0.0/ows19115subset.xsd">
```

```
  <Coverage>
    <ows:Title>Example</ows:Title>
    <ows:Abstract>Example of coverage</ows:Abstract>
    <Identifier>Example</Identifier>
    <Reference href="cid:bfa8e8ac@example.net" />
    <Reference href="cid:c3499c27@example.net" />
    <Reference href="http://example.net/data/data2.nc" />
  </Coverage>
</Coverages>
------=_NextPart_000_0000_93251752.3C5526C0
Content-Type: application/xml
Content-ID: <bfa8e8ac@example.net>

...(metadata content)...
------=_NextPart_000_0000_93251752.3C5526C0
Content-Type: application/CF-netCDF3
Content-Transfer-Encoding: base64
Content-ID: <c3499c27@example.net>

...(1st data content in netCDF)...
------=_NextPart_000_0000_93251752.3C5526C0
Content-Type: application/CF-netCDF3
Content-Transfer-Encoding: base64
Content-ID: <7904e041@example.net>
Content-Location: http://example.net/data/data2.nc

...(2nd data content in netCDF)...
------=_NextPart_000_0000_93251752.3C5526C0--
```

C.3.3 SOAP Response with binary and ncML data

```
MIME-Version: 1.0

Content-Type: multipart/related; boundary="----=_NextPart_000_0000_93251752.3C5526C0"



------=_NextPart_000_0000_93251752.3C5526C0

Content-Type: application/xml; charset="UTF-8"



<?xml version="1.0" encoding="UTF-8"?>

<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope">

  <soap:Header/>

  <soap:Body>

    <Coverages xmlns="http://www.opengis.net/wcs/1.1/ows" xmlns:ows="http://www.opengis.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:schemaLocation="http://www.opengis.net/wcs/1.1/ows ../owsCoverages.xsd
http://www.opengis.net/ows ../../../ows/1.0.0/ows19115subset.xsd">

      <Coverage>

        <ows:Title>Example</ows:Title>

        <ows:Abstract>Example of coverage</ows:Abstract>

        <Identifier>Example</Identifier>

        <Reference href="cid:bfa8e8ac@example.net" />

        <Reference href="cid:c3499c27@example.net" />

        <Reference href="http://example.net/data/data2.ncml" />

      </Coverage>

    </Coverages>

  </soap:Body>

</soap:Envelope>

------=_NextPart_000_0000_93251752.3C5526C0

Content-Type: application/xml

Content-ID: <bfa8e8ac@example.net>



...(metadata content)...
```

```
------=_NextPart_000_0000_93251752.3C5526C0
Content-Type: application/CF-netCDF3
Content-Transfer-Encoding: base64
Content-ID: <c3499c27@example.net>

...(data content in netCDF)...
------=_NextPart_000_0000_93251752.3C5526C0
Content-Type: application/CF-netCDF3
Content-Transfer-Encoding: base64
Content-ID: <1324bc12@example.net>
Content-Location: http://example.net/data/data2.ncml

...(data content in ncML)...
------=_NextPart_000_0000_93251752.3C5526C0--
```

C.3.4 Multipart section containing ncML with binary data included

```
------=_NextPart_000_0000_93251752.3C5526C0
Content-Type: application/CF-netCDF3
Content-Transfer-Encoding: base64
Content-ID: <1324bc12@example.net>
Content-Location: http://example.net/data/data2.ncml

<?xml version="1.0" encoding="UTF-8"?>
<netcdf xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://www.unidata.ucar.edu/namespaces/netcdf/ncml-2.2">
   <dimension name="longitude" length="96"/>
   <dimension name="bounds_axis" length="2"/>
   <dimension name="latitude" length="73"/>
   <dimension name="air_pressure" length="15"/>
   <dimension name="time" length="140"/>
```

```
<attribute name="Conventions" type="string" value="CF-1.1"/>
<attribute name="source" type="string" value="Data from model run ABC123"/>
<variable name="longitude" shape="longitude" type="double">
   <attribute name="standard_name" type="string" value="longitude"/>
   <attribute name="units" type="string" value="degrees_east"/>
   <attribute name="bounds" type="string" value="bound_longitude"/>
   <attribute name="axis" type="string" value="X"/>
   <values increment="3.75" npts="96" start="0.0"/>
</variable>
<variable name="bound_longitude" shape="longitude bounds_axis" type="double"> </variable>
<variable name="latitude" shape="latitude" type="double">
   <attribute name="standard_name" type="string" value="latitude"/>
   <attribute name="units" type="string" value="degrees_north"/>
   <attribute name="bounds" type="string" value="bound_latitude"/>
   <attribute name="axis" type="string" value="Y"/>
   <values increment="-2.5" npts="73" start="90.0"/>
</variable>
<variable name="bound_latitude" shape="latitude bounds_axis" type="double"> </variable>
<variable name="air_pressure" shape="air_pressure" type="float">
   <attribute name="standard_name" type="string" value="air_pressure"/>
   <attribute name="units" type="string" value="hPa"/>
   <attribute name="axis" type="string" value="Z"/>
   <attribute name="positive" type="string" value="down"/>
   <values increment="70.71429" npts="15" start="10.0"/>
</variable>
<variable name="time" shape="time" type="double">
   <attribute name="calendar" type="string" value="360_day"/>
   <attribute name="standard_name" type="string" value="time"/>
   <attribute name="units" type="string" value="days since 2289-1-1"/>
   <attribute name="bounds" type="string" value="bound_time"/>
```

```
      <attribute name="axis" type="string" value="T"/>
      <values increment="360.0" npts="140" start="510.0"/>
   </variable>
   <variable name="bound_time" shape="time bounds_axis" type="double"> </variable>
   <variable name="air_temperature" shape="time air_pressure latitude longitude" type="float">
      <attribute name="standard_name" type="string" value="air_temperature"/>
      <attribute name="units" type="string" value="Kir_temperature"/>
      <attribute name="long_name" type="string" value="temperature on pressure level"/>
      <attribute name="cell_methods" type="string" value="longitude: latitude: mean time: mean (interval:
4 h)"/>
      <attribute name="_FillValue" type="float" value="-1.073742e+09"/>
      <values xm:xmlmime="binary/octet-
stream">Li4uKGJhc2U2NCBlbmNvZGVkIGRhdGEpLi4u...(base64 encoded
data)...BlbmNvZGVkIGRhdGEpLi4u==</values>
   </variable>
</netcdf>
------=_NextPart_000_0000_93251752.3C5526C0--
```

C.3.5 Multipart section containing ncML with binary data extracted using XOP

```
------=_NextPart_000_0000_93251752.3C5526C0
MIME-Version: 1.0
Content-Type: Multipart/Related;boundary=MIME_boundary;  type="application/xop+xml";
   start="<12dea45c@example.net >";
   startinfo="application/ncML+xml;


--MIME_boundary
Content-Type: application/xop+xml; charset=UTF-8;
   type="application/ncML+xml; action=\"ProcessData\""
Content-ID: <12dea45c@example.net>
Content-Location: http://example.net/data/data2.ncml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<netcdf xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://www.unidata.ucar.edu/namespaces/netcdf/ncml-2.2">
   <dimension name="longitude" length="96"/>
   <dimension name="bounds_axis" length="2"/>
   <dimension name="latitude" length="73"/>
   <dimension name="air_pressure" length="15"/>
   <dimension name="time" length="140"/>
   <attribute name="Conventions" type="string" value="CF-1.1"/>
   <attribute name="source" type="string" value="Data from model run ABC123"/>
   <variable name="longitude" shape="longitude" type="double">
      <attribute name="standard_name" type="string" value="longitude"/>
      <attribute name="units" type="string" value="degrees_east"/>
      <attribute name="bounds" type="string" value="bound_longitude"/>
      <attribute name="axis" type="string" value="X"/>
      <values increment="3.75" npts="96" start="0.0"/>
   </variable>
   <variable name="bound_longitude" shape="longitude bounds_axis" type="double"> </variable>
   <variable name="latitude" shape="latitude" type="double">
      <attribute name="standard_name" type="string" value="latitude"/>
      <attribute name="units" type="string" value="degrees_north"/>
      <attribute name="bounds" type="string" value="bound_latitude"/>
      <attribute name="axis" type="string" value="Y"/>
      <values increment="-2.5" npts="73" start="90.0"/>
   </variable>
   <variable name="bound_latitude" shape="latitude bounds_axis" type="double"> </variable>
   <variable name="air_pressure" shape="air_pressure" type="float">
      <attribute name="standard_name" type="string" value="air_pressure"/>
      <attribute name="units" type="string" value="hPa"/>
      <attribute name="axis" type="string" value="Z"/>
```

```
      <attribute name="positive" type="string" value="down"/>
      <values increment="70.71429" npts="15" start="10.0"/>
   </variable>
   <variable name="time" shape="time" type="double">
      <attribute name="calendar" type="string" value="360_day"/>
      <attribute name="standard_name" type="string" value="time"/>
      <attribute name="units" type="string" value="days since 2289-1-1"/>
      <attribute name="bounds" type="string" value="bound_time"/>
      <attribute name="axis" type="string" value="T"/>
      <values increment="360.0" npts="140" start="510.0"/>
   </variable>
   <variable name="bound_time" shape="time bounds_axis" type="double"> </variable>
   <variable name="air_temperature" shape="time air_pressure latitude longitude" type="float">
      <attribute name="standard_name" type="string" value="air_temperature"/>
      <attribute name="units" type="string" value="Kir_temperature"/>
      <attribute name="long_name" type="string" value="temperature on pressure level"/>
      <attribute name="cell_methods" type="string" value="longitude: latitude: mean time: mean (interval:
4 h)"/>
      <attribute name="_FillValue" type="float" value="-1.073742e+09"/>
      <values xm:xmlmime="binary/octet-stream"><xop:Include
   xmlns:xop='http://www.w3.org/2004/08/xop/include'
   href='cid:bd43gh2y@example.net'/>
      </values>
   </variable>
</netcdf>
--MIME_boundary
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary
Content-ID: <bd43gh2y@example.net>
```

```
...(binary encoded data)...

--MIME_boundary--

------=_NextPart_000_0000_93251752.3C5526C0--
```

Appendix D: Compliance Testing

This chapter contains pointers to mechanisms for testing whether resulting WCS coverages conform to the encoding format.

(see CITE)

TBD