Open Geospatial Consortium

Date: 2011-04-04

Reference number of this OGC® project document: **OGC 10-092r3**

OGC name of this OGC® project document: **http://www.opengis.net/doc/IS/netcdf-binary/1.0**

Version: 1.0

Category: OGC® Candidate Encoding Standard

Editor: Ben Domenico

NetCDF Binary Encoding Extension Standard: NetCDF Classic and 64-bit Offset Format

Warning

This document is an OGC Member approved international standard. This document is available on a royalty free, non-discriminatory basis. Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type:	OGC® Implementation Standard
Document subtype:	Encoding
Document stage:	Approved
Document language:	English

Table of Contents

i. Abstract

This document defines an OGC® Standard for encoding binary representations of space-time varying geo-referenced data. Specifically, this standard specifies the netCDF classic and 64-bit offset file binary encoding formats. This standard specifies a set of requirements that every netCDF classic or 64-bit offset binary encoding must fulfil.

ii. Keywords

ogcdoc, netcdf, space-time, netcdf-classic

iii. Preface

This is an OGC® Standard for encoding binary representations of space-time varying geo-referenced data.

iv. Document Terms and definitions

This document uses the specification terms defined in Subclause 5.3 of [OGC 06-121r8], 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.

v. Submitting organizations

The following organizations submitted this Candidate Implementation Specification to the Open Geospatial Consortium Inc.

- IMAA-CNR Italy
- METEO-FRANCE

- Natural Environment Research Council (NERC)
- Northrop Grumman Corporation
- University Corporation for Atmospheric Research (UCAR)
- US National Oceanic and Atmospheric Administration (NOAA)

vi. Submission contact points

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

CONTACT	COMPANY
Ben Domenico, editor	Unidata Program Center, UCAR
Russ Rew	Unidata Program Center, UCAR
Ethan Davis, Dennis Heimbigner, Ed Hartnett John Caron	Unidata Program Center, UCAR

vii. Changes to the OGC® Abstract Specification

The OGC® Abstract Specification does not require changes to accommodate this OGC® standard.

Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights. However, to date, no such rights have been claimed or identified.

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 specification set forth in this document, and to provide supporting documentation.

Introduction

NetCDF (Network Common Data Form) is a data model for array-oriented scientific data. There is a freely distributed collection of access libraries implementing support for that data model, and a machine-independent format. Together, the interfaces, libraries, and format support the creation, access, and sharing of scientific data.

Background information regarding the overall landscape of netCDF standards is presented in the CF-netCDF Primer, OGC 10-091r3, "CF-netCDF: Core and Extensions." This standard is an extension to the core specification for the netCDF Classic data model in OGC 10-091r3, "NetCDF Core."

OGC Binary Encoding Extension Standard: netCDF Classic and 64-bit Offset Format

1 Scope

This standard specifies the netCDF classic and 64-bit offset file binary encoding formats. This standard specifies a set of requirements that every netCDF classic or 64-bit offset binary encoding must fulfil.

2 Conformance

Standardization targets are netCDF classic and 64-bit offset binary dataset encodings.

This document establishes three conformance classes of:

- *netCDF common* with URI http://www.opengis.net/spec/netcdf-binary/1.0/conf/common

- *netCDF classic* with URI http://www.opengis.net/spec/netcdf-binary/1.0/conf/classic

- *netCDF 64-bit offset* with URI http://www.opengis.net/spec/netcdf-binary/1.0/conf/64-bit-offset

Requirements and conformance test URIs defined in this document are relative to http://www.opengis.net/spec/netcdf-binary/1.0.

Annex A (normative) specifies how data are encoded in netCDF classic and 64-bit offset binary formats. In addition, these encodings must satisfy the tests listed the abstract test suite for the netCDF core in Annex A of OGC 10-090r3, “NetCDF Core.”

3 Normative references

The *NetCDF Classic and 64-bit Binary Encoding Extension* is contained within this document. The specification is identified by OGC URI http://www.opengis.net/spec/netcdf-binary/1.0.

The document has OGC URL http://www.opengis.net/doc/IS/netcdf-binary/1.0.

The following normative document contains provisions that, through reference in this text, constitute provisions of this specification. For undated references, the latest edition of the referenced document (including any amendments) applies.

NetCDF Core Specification. OGC Document 10-091r3.
http://www.opengis.net/doc/IS/netcdf

For this specification, there is one external normative document contain provisions that are quoted verbatim in this text and hence constitute provisions of this specification.

NASA ESDS-RFC-011v2.00 R. Rew, E. Hartnett, D. Heimbigner, E. Davis, J. Caron: *NetCDF Classic and 64-bit Offset File Formats*

http://www.esdswg.org/spg/rfc/esds-rfc-011/ESDS-RFC-011v2.00.pdf

4 Terms and definitions

4.1 Definitions

For purposes of this document, the definitions in OGC 10-090r3, NetCDF Core, apply.

4.2 Acronyms (and abbreviated terms)

Some frequently used abbreviated terms:

API	Application Program Interface
BNF	Backus-Naur Form
CF	Climate and Forecast Conventions
ESDSWG	NASA Earth Standards Data Systems Working Groups
ES	Earth Sciences
GIS	Geographic Information System
HDF5	Hierarchical Data Format version 5
NcML	NetCDF Markup Language
NcML-GML	NetCDF Markup Language – Geography Markup Language
NetCDF	Network Common Data Form
NetCDF-4	NetCDF Release 4
ISO	International Organization for Standardization
OGC	Open Geospatial Consortium
SPG	NASA Standards Process Group

UML	Unified Modeling Language
XML	eXtended Markup Language
WFS	Web Feature Service
WCS	Web Coverage Service
1-D	One Dimensional
2-D	Two Dimensional

5 Document Conventions

5.1 UML Notation

The diagrams that appear in this standard are presented using the Unified Modeling Language (UML) static structure diagram.

5.2 BNF Notation

To present the format more formally, we use a BNF grammar notation. In this notation:

- Non-terminals (entities defined by grammar rules) are in lower case.
- Terminals (atomic entities in terms of which the format specification is written) are in upper case, and are specified literally as US-ASCII characters within single-quote characters or are described with text between angle brackets (‘<’ and ‘>’).
- Optional entities are enclosed between braces (‘[’ and ‘]’).
- A sequence of zero or more occurrences of an entity is denoted by ‘[entity ...]’.
- A vertical line character (‘|’) separates alternatives. Alternation has lower precedence than concatenation.
- Comments follow ‘//’ characters.
- A single byte that is not a printable character is denoted using a hexadecimal number with the notation ‘\xDD’, where each D is a hexadecimal digit.
- A literal single-quote character is denoted by ‘\'’, and a literal back-slash character is denoted by ‘\\’.

Following the grammar, a few additional notes are included to specify format characteristics that are impractical to capture in a BNF grammar, and to note some special cases for implementers. Comments in the grammar point to the notes and special cases, and help to clarify the intent of elements of the format.

5.3 Namespace prefix conventions

Since there are no XML schemas used in this standard, there are no namespace mappings

6 netCDF Classic and 64-bit Offset File Formats Extension Standard

This document formally specifies two format variants, the classic binary format and the 64-bit offset format for netCDF data. The NetCDF Classic Data Model is specified in OGC 10-NCD, "NetCDF Core."

Following are the requirements for the netCDF classic and 64-bit offset file format. Understanding the format at this level can make clear which netCDF operations are expensive, for example adding a new variable to an existing file.

This standard defines two levels of conformance, as shown in the following diagram.

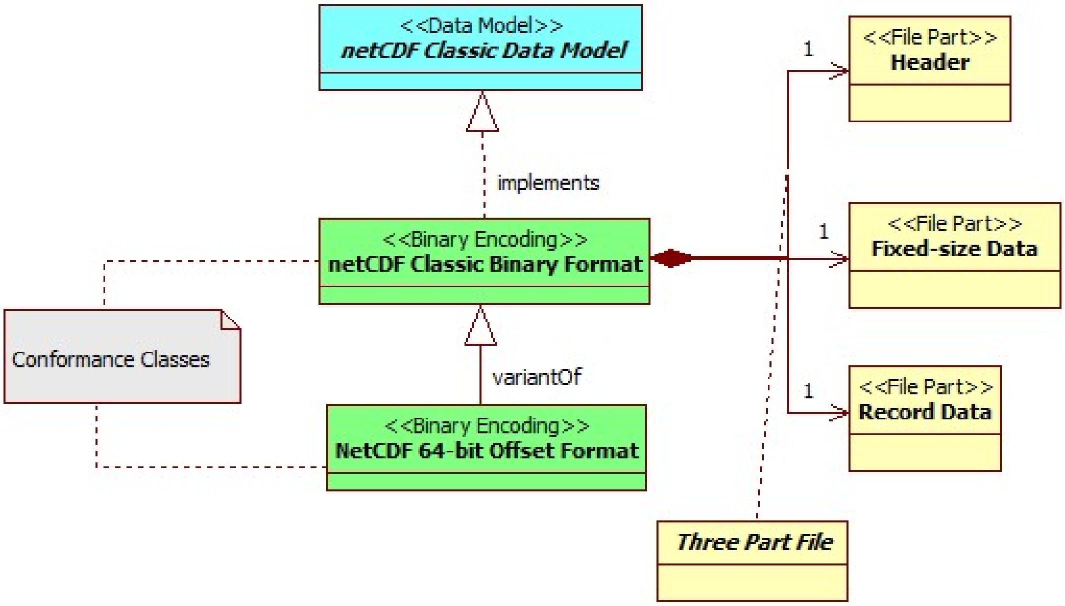

Figure 1 Conformance classes and modules diagram

Referring to Figure 1, the following conformance classes and modules are defined, as detailed in the following paragraphs. Related conformance test cases are defined in Annex A.

Most elements of the encoding are common to both the classic binary and 64-bit offset variants, so there is a common conformance class. NetCDF classic encodings must satisfy the tests of the common class as well as those of the classic binary class. NetCDF 64-bit-offset encodings must satisfy the tests of the common class as well as those of the 64-bit-offset class.

NetCDF Classic Data Model (This is a conformance module for both the two conformance classes) http://www.opengis.net/spec/netcdf/1.0/conf/core	certifies the conformance to the abstract netCDF data model (array-oriented scientific data) requirement OGC 10-091r3, “NetCDF Core.”
NetCDF Common Binary (This is a conformance class) http://www.opengis.net/spec/netcdf-binary/1.0/conf/common	certifies the conformance to the common binary format requirements of 6.1.2
NetCDF Classic Binary (This is a conformance class) http://www.opengis.net/spec/netcdf-binary/1.0/conf/classic	certifies the conformance to the classic binary format requirement of 6.1.3
NetCDF Binary-64-bit-Offset Format (This is a conformance class) http://www.opengis.net/spec/netcdf-binary/1.0/conf/64-bit-offset	certifies the conformance to the 64-bit variant binary format requirements of 6.1.4
Three Part File (This is a conformance module for the common conformance class)	certifies the requirement for a three part file requirements of 6.1.2.1
Header (This is a conformance module for the common conformance class)	certifies the conformance to the netCDF header requirements of 6.1.2.2
Fixed-size (non-record) Data (This is a conformance module for the common conformance class)	certifies the conformance to the netCDF fixed-size data requirements of 6.1.2.3
Record Data (This is a conformance module for the common conformance class)	certifies the conformance to the netCDF record data requirements of 6.1.2.4

Table 1 Conformance class table

6.1.1 NetCDF Classic Abstract Data Model

Requirement 1 http://www.opengis.net/spec/netcdf-binary/1.0/req/common/data-model

The data shall conform to the netCDF classic abstract model as specified in the document OGC 10-NCD, "NetCDF Core." Related conformance test cases are defined in Annex A of OGC 10-NCD, "NetCDF Core."

6.1.2 NetCDF Binary Dataset Format: Common Elements

The data shall conform to the netCDF binary file format as specified in the following sections. Related conformance test cases are defined in section A.1.

6.1.2.1 Three Part File

A classic or 64-bit offset file shall be stored in three parts: the header, the fixed-size (non-record) data, and the record data. Related conformance test cases are defined in section A.1.

Requirement 2 http://www.opengis.net/spec/netcdf-binary/1.0/req/common/netcdf-dataset-components/

A netCDF dataset shall have a header (*header*) section and a data (*data*) section.

Requirement 3 http://www.opengis.net/spec/netcdf-binary/1.0/req/common/data-section-components/

The data section shall have a fixed –size, non-record (*non-recs*) and record (*recs)* section

Requirement 4 http://www.opengis.net/spec/netcdf-binary/1.0/req/common/header-part/

There shall be only one header part per file.

Requirement 5 http://www.opengis.net/spec/netcdf-binary/1.0/req/common/fixed-size-data-part/

There shall be only one fixed-size data part per file.

Requirement 6 http://www.opengis.net/spec/netcdf-binary/1.0/req/common/record-data-part/

There shall be only one record data part per file.

Requirement 7 http://www.opengis.net/spec/netcdf-binary/1.0/req/common/BNF-for-header-non-recore-record/

The header, non-record and record parts shall conform to the BNF grammar segment given below

```
netcdf_file  = header data

   .

   .

   .

data         = non_recs recs
```

6.1.2.2 The Header

Requirement 8 http://www.opengis.net/spec/netcdf-binary/1.0/req/common/header-part-specifications/

The header shall specify:

- whether classic or 64-bit offset encoding (*magic)* is used
- the length of the record dimension (*numrecs*)
- the list of dimenisons (*dim_list*)
- the list of global attributes (*gatt_list*)
- the list of variables (*var_list*)

Requirement 9 http://www.opengis.net/spec/netcdf-binary/1.0/req/common/BNF-for-header/

The header shall conform to the BNF grammar segment given below.

```
header       = magic numrecs dim_list gatt_list var_list
magic        = 'C' 'D' 'F' VERSION
VERSION      = \x01 |                  // classic format
               \x02                    // 64-bit offset format
numrecs      = NON_NEG | STREAMING     // length of record dimension
dim_list     = ABSENT | NC_DIMENSION nelems [dim ...]
gatt_list    = att_list                // global attributes
att_list     = ABSENT | NC_ATTRIBUTE nelems [attr ...]
var_list     = ABSENT | NC_VARIABLE nelems [var ...]
ABSENT       = ZERO ZERO               // Means list is not present
ZERO         = \x00 \x00 \x00 \x00     // 32-bit zero
NC_DIMENSION = \x00 \x00 \x00 \x0A     // tag for list of dimensions
NC_VARIABLE  = \x00 \x00 \x00 \x0B     // tag for list of variables
NC_ATTRIBUTE = \x00 \x00 \x00 \x0C     // tag for list of attributes
nelems       = NON_NEG                 // number of elements in following sequence
dim          = name dim_length
```

```
name         = nelems namestring
                   // Names a dimension, variable, or attribute.
                   // Names should match the regular expression
                   //([a-zA-Z0-9_]|{MUTF8})([^\x00-\x1F/\x7F-\xFF]|{MUTF8})*
                   // For other constraints, see “Note on names”, below.
namestring   = ID1 [IDN ...] padding
ID1          = alphanumeric | '_'
IDN          = alphanumeric | special1 | special2
alphanumeric = lowercase | uppercase | numeric | MUTF8
lowercase    = 'a'|'b'|'c'|'d'|'e'|'f'|'g'|'h'|'i'|'j'|'k'|'l'|'m'|
               'n'|'o'|'p'|'q'|'r'|'s'|'t'|'u'|'v'|'w'|'x'|'y'|'z'
uppercase    = 'A'|'B'|'C'|'D'|'E'|'F'|'G'|'H'|'I'|'J'|'K'|'L'|'M'|
               'N'|'O'|'P'|'Q'|'R'|'S'|'T'|'U'|'V'|'W'|'X'|'Y'|'Z'
numeric      = '0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9'




                    // special1 chars have traditionally been
                    // permitted in netCDF names.
special1     = '_'|'.'|'@'|'+'|'-'
                    // special2 chars are recently permitted in
                    // names (and require escaping in CDL).
                    // Note: '/' is not permitted.
special2     = ' ' | '!' | '"' | '#'  | '$' | '%' | '&' | '\'' |
               '(' | ')' | '*' | ','  | ':' | ';' | '<' | '='  |
               '>' | '?' | '[' | '\\' | ']' | '^' | '`' | '{'  |
               '|' | '}' | '~'
MUTF8        = <multibyte UTF-8 encoded, NFC-normalized Unicode character>
dim_length   = NON_NEG       // If zero, this is the record dimension.
                             // There can be at most one record dimension.
attr         = name nc_type nelems [values ...]
nc_type      = NC_BYTE | NC_CHAR | NC_SHORT | NC_INT | NC_FLOAT | NC_DOUBLE
var          = name nelems [dimid ...] vatt_list nc_type vsize begin
                             // nelems is the dimensionality (rank) of the
                             // variable: 0 for scalar, 1 for vector, 2
                             // for matrix, ...
dimid        = NON_NEG       // Dimension ID (index into dim_list) for
                             // variable shape. We say this is a "record
                             // variable" if and only if the first
                             // dimension is the record dimension.
vatt_list    = att_list      // Variable-specific attributes
vsize        = NON_NEG       // Variable size. If not a record variable,
                             // the amount of space in bytes allocated to
                             // the variable's data. If a record variable,
                             // the amount of space per record. See “Note on
                             // vsize” below.
begin        = OFFSET        // Variable start location. The offset in
                             // bytes (seek index) in the file of the
                             // beginning of data for this variable.
```

6.1.2.3 The Fixed-size (Non-record) Data

Requirement 10 http://www.opengis.net/spec/netcdf-binary/1.0/req/common/contiguous-for-fixed-size-data

The data for all non-record variables shall be stored contiguously for each variable, in the same order the variables occur in the header.

Requirement 11 http://www.opengis.net/spec/netcdf-binary/1.0/req/common/block-values-for-fixed-size-data

All data for a non-record variable shall be stored as a block of values of the same type as the variable, in row-major order (last dimension varying fastest).

Requirement 12 http://www.opengis.net/spec/netcdf-binary/1.0/req/common/data-values-for non-record-variable

The fixed-size data shall contain data values for variables that don't have an unlimited dimension, i.e., for each non-record variable.

Requirement 13 http://www.opengis.net/spec/netcdf-binary/1.0/req/common/data-row-major

The data for each variable shall be stored contiguously, in row-major order for multi-dimensional variables.

Requirement 14 http://www.opengis.net/spec/netcdf-binary/1.0/req/common/BNF-for-fixed-size-data

The fixed-size (non-record) data shall conform to the BNF grammar segment given below.

```
non_recs     = [vardata ...] // The data for all non-record variables,
                             // stored contiguously for each variable, in
                             // the same order the variables occur in the
                             // header.
vardata      = [values ...]  // All data for a non-record variable, as a
                             // block of values of the same type as the


                             // variable, in row-major order (last
                             // dimension varying fastest).
```

Related conformance test cases are defined in section A.1.

6.1.2.4 The Record Data

Requirement 15 http://www.opengis.net/spec/netcdf-binary/1.0/req/common/one-unlimited-dimension

There shall be at most one unlimited dimension, the record dimension.

Requirement 16 http://www.opengis.net/spec/netcdf-binary/1.0/req/common/data-values-for-unlimited-dimension

The record data shall contain data values for variables that have an unlimited dimension.

Requirement 17 http://www.opengis.net/spec/netcdf-binary/1.0/req/common/current-size-in-header

The current size of the record dimension shall be stored in the header which specifies how many records the file contains.

Requirement 18 http://www.opengis.net/spec/netcdf-binary/1.0/req/common/all-data-for-record-part

Each record in the record data part shall contain all the data for that record for each record variable.

Requirement 19 http://www.opengis.net/spec/netcdf-binary/1.0/req/common/data-for-each-record

Each record's worth of data for each record variable shall be stored contiguously, in row major order for multidimensional variables.

Requirement 20 http://www.opengis.net/spec/netcdf-binary/1.0/req/common/record-size

All records shall be the same size, because they each contain all the data for a particular record for each record variable.

Requirement 21 http://www.opengis.net/spec/netcdf-binary/1.0/req/common/BNF-for-record-data

The record data section shall conform to the BNF grammar segment given below.

```
recs         = [record ...]  // The data for all record variables are
                             // stored interleaved at the end of the
                             // file.
record       = [varslab ...] // Each record consists of the n-th slab
                             // from each record variable, for example
                             // x[n,...], y[n,...], z[n,...] where the
                             // first index is the record number, which
                             // is the unlimited dimension index.
varslab      = [values ...]  // One record of data for a variable, a
                             // block of values all of the same type as
                             // the variable in row-major order (last
                             // index varying fastest).
```

Related conformance test cases are defined in section A.1.

6.1.2.5 BNF Definitions

Requirement 22 http://www.opengis.net/spec/netcdf-binary/1.0/req/common/BNF-definitions

The BNF segments in the previous requirements shall conform to the BNF specifications in the segment below.

```
values       = bytes | chars | shorts | ints | floats | doubles
string       = nelems [chars]
bytes        = [BYTE ...] padding
chars        = [CHAR ...] padding
shorts       = [SHORT ...] padding
ints         = [INT ...]
floats       = [FLOAT ...]
doubles      = [DOUBLE ...]
padding      = <0, 1, 2, or 3 bytes to next 4-byte boundary>
                           // Header padding uses null (\x00) bytes. In
                           // data, padding uses variable's fill value.
                           // See “Note on padding” below for a special
                           // case.
NON_NEG      = <non-negative INT>
STREAMING    = \xFF \xFF \xFF \xFF   // Indicates indeterminate record
                                     // count, allows streaming data
OFFSET       = <non-negative INT> |  // For classic format
               <non-negative INT64>  // for 64-bit offset format
BYTE         = <8-bit byte>          // See “Note on byte data”, below.
CHAR         = <8-bit byte>          // See “Note on char data”, below.
SHORT        = <16-bit signed integer, Bigendian, two's complement>
INT          = <32-bit signed integer, Bigendian, two's complement>
INT64        = <64-bit signed integer, Bigendian, two's complement>
FLOAT        = <32-bit IEEE single-precision float, Bigendian>
DOUBLE       = <64-bit IEEE double-precision float, Bigendian>
                           // following type tags are 32-bit integers
NC_BYTE      = \x00 \x00 \x00 \x01       // 8-bit signed integers
NC_CHAR      = \x00 \x00 \x00 \x02       // text characters
NC_SHORT     = \x00 \x00 \x00 \x03       // 16-bit signed integers
NC_INT       = \x00 \x00 \x00 \x04       // 32-bit signed integers
NC_FLOAT     = \x00 \x00 \x00 \x05       // IEEE single precision floats
NC_DOUBLE    = \x00 \x00 \x00 \x06       // IEEE double precision floats
                           // Default fill values for each type, may be
                           // overridden by variable attribute named
                           // ‘_FillValue’, see “Note on fill values”, below
FILL_BYTE    = \x81                      // (signed char) -127
FILL_CHAR    = \x00                      // null byte
FILL_SHORT   = \x80 \x01                 // (short) -32767
FILL_INT     = \x80 \x00 \x00 \x01       // (int) -2147483647
FILL_FLOAT   = \x7C \xF0 \x00 \x00           // (float) 9.9692099683868690e+36
FILL_DOUBLE  = \x47 \x9E \x00 \x00 \x00 \x00 // (double)9.9692099683868690e+36
```

6.1.3 NetCDF Classic Variant

The netCDF classic format specifies the VERSION byte as \x01, and the OFFSET entity as a 32-bit offset from the beginning of the file. Related conformance test cases are defined in section A.2.

Requirement 23 http://www.opengis.net/spec/netcdf-binary/1.0/req/classic/Version-Offset

A netCDF classic dataset shall conform to all the requirements of the netCDF binary encoding with the the BNF grammar values for VERSION and OFFSET as given below.

```
VERSION      =  \x01                   // classic format


OFFSET       =  <non-negative INT>     // classic format
```

6.1.4 NetCDF 64-bit Offset Variant

The netCDF 64-bit offset format differs from the classic format only in the VERSION byte, \x02 instead of \x01, and the OFFSET entity, a 64-bit instead of a 32-bit offset from the beginning of the file. Related conformance test cases are defined in section A.3.

Requirement 24 http://www.opengis.net/spec/netcdf-binary/1.0/req/64-bit-offset/Version-Offset

A netCDF 64-bit Offset Variant dataset shall conform to all the requirements of the netCDF binary encoding with the the BNF grammar values for VERSION and OFFSET as given below.

```
VERSION      =  \x02                   // 64-bit offset format


OFFSET       =  <non-negative INT64>  // for 64-bit offset format
```

This small format change permits much larger files, but there are still some practical size restrictions. Each fixed-size variable and the data for one record's worth of each record variable are still limited in size to a little less that 4 GiB. The rationale for this limitation is to permit aggregate access to all the data in a netCDF variable (or a record's worth of data) on 32-bit platforms.

6.1.5 BNF Supplementary Notes

The following notes apply to the BNF segments in the specifications above.

Note on vsize: This number is the product of the dimension lengths (omitting the record dimension) and the number of bytes per value (determined from the type), increased to the next multiple of 4, for each variable. If a record variable, this is the amount of space per record. The netCDF “record size” is calculated as the sum of the vsize's of all the record variables.

The vsize field is actually redundant, because its value may be computed from other information in the header. The 32-bit vsize field is not large enough to contain the size of variables that require more than 2^{32} - 4 bytes, so 2^{32} - 1 is used in the vsize field for such variables.

Note on names: Earlier versions of the netCDF C-library reference implementation enforced a more restricted set of characters in creating new names, but permitted reading names containing arbitrary bytes. This RFC extends the permitted characters in names to include multi-byte UTF-8 encoded[7] Unicode[4] and additional printing characters from the US-ASCII alphabet. The first character of a name must be alphanumeric, a multi-byte UTF-8 character, or '_' (traditionally reserved for names with meaning to implementations, such as the "_FillValue" attribute). Subsequent characters may also include printing special characters, except for '/' which is not allowed in names. Names that have trailing space characters are also not permitted.

Implementations of the netCDF classic and 64-bit offset format must ensure that names are normalized according to Unicode NFC normalization rules [5] during encoding as UTF-8 for storing in the file header. This is necessary to ensure that gratuitous differences in the representation of Unicode names do not cause anomalies in comparing files and querying data objects by name.

Note on streaming data: The largest possible record count, 2^{32}-1, is reserved to indicate an indeterminate number of records. This means that the number of records in the file must be determined by other means, such as reading them or computing the current number of records from the file length and other information in the header. It also means that the numrecs□field in the header will not be updated as records are added to the file.

Note on padding: In the special case of only a single record variable of character, byte, or short type, no padding is used between data values.

Note on byte data: It is possible to interpret byte data as either signed (-128 to 127) or unsigned (0 to 255). When reading byte data through an interface that converts it into another numeric type, the default interpretation is signed. There are various attribute conventions for specifying whether bytes represent signed or unsigned data, but no standard convention has been established. The variable attribute "_Unsigned" is reserved for this purpose in future implementations.

Note on char data: Although the characters used in netCDF names must be encoded as UTF-8, character data may use other encodings. The variable attribute "_Encoding" is reserved for this purpose in future implementations.

Note on fill values: Because data variables may be created before their values are written, and because values need not be written sequentially in a netCDF file, default "fill values" are defined for each type, for initializing data values before they are explicitly written. This makes it possible to detect reading values that were never written. The variable attribute "_FillValue", if present, overrides the default fill value for a variable. If _FillValue is defined then it should be scalar and of the same type as the variable.

References

NetCDF Climate and Forecast (CF) Metadata Conventions
http://www.cfconventions.org/ or http://cf-pcmdi.llnl.gov/

Unidata UCAR, NetCDF User Guide
http://www.unidata.ucar.edu/netcdf/docs/netcdf.html

Unidata UCAR, NetCDF Reference Implementations
ftp://ftp.unidata.ucar.edu/pub/netcdf/netcdf.tar.

NetCDF C Language Interface Guide
http://www.unidata.ucar.edu/netcdf/docs/netcdf-c/

NetCDF C++ Language Interface Guide
http://www.unidata.ucar.edu/netcdf/docs/netcdf-cxx/

NetCDF FORTRAN Language Interface Guides
http://www.unidata.ucar.edu/netcdf/docs/netcdf-f77/
http://www.unidata.ucar.edu/netcdf/docs/netcdf-f90/
NetCDF Java Language Interface Guide
http://www.unidata.ucar.edu/netcdf-java/

IETF RFC 2616, Hypertext Transfer Protocol – HTTP/1.1. (June 1999)

ISO 8601:2004, Data elements and interchange formats — Information interchange — Representation of dates and times.

ISO 19101:2002, Geographic information — Reference model

ISO 19107:2003, Geographic Information — Spatial schema

ISO 19111:—1), Geographic Information — Spatial referencing by coordinates

ISO 19123, Abstract Coverage Specification

ISO 19136:2007, Geographic information — Geography Markup Language (GML)

OGC 00-014r1, Guidelines for Successful OGC Interface Specification

Annex A: Conformance Class Abstract Test Suite (Normative)

A.1 Conformance Test Class: netCDF Binary Common

Applies for conformance class netCDF binary with URI
http://www.opengis.net/spec/netcdf-binary/1.0/conf/common

A.1.1 Requirement 1

Test ID	**Conformance Test for Requirement 1** **http://www.opengis.net/spec/netcdf-binary/1.0/conf/common/netcdf-data-model**
Test purpose	The data shall conform to the netCDF classic abstract model as specified in OGC 10-NCD, “NetCDF Core.” Related conformance test cases are defined in Annex A of OGC 10-NCD, “NetCDF Core.”
Test method	Verify that dataset satisfies the core conformance test cases defined in Annex A of OGC 10-NCD, “NetCDF Core.”

A.1.2 Requirement 2

Test ID	**Conformance Test for Requirement 2** **http://www.opengis.net/spec/netcdf-binary/1.0/conf/common/netcdf-dataset-components**
Test purpose	A netCDF dataset shall have a header (*header*) section and a data (*data*) section.
Test method	Open the dataset and verify that it has a header (*header*) section and a data (*data*) section.

A.1.3 Requirement 3

Test ID	**Conformance Test for Requirement 3** **http://www.opengis.net/spec/netcdf-binary/1.0/conf/common/data-section-components**

Test purpose	The data section shall have a fixed –size, non-record (*non-recs*) and record (*recs)* section
Test method	Open the dataset and verify that it has a fixed –size, non-record (*non-recs*) and record (*recs)* section.

A.1.4 Requirement 4

Test ID	**Conformance Test for Requirement 4** **http://www.opengis.net/spec/netcdf-binary/1.0/conf/common/header-part**
Test purpose	There shall be only one header part per file
Test method	Open the dataset and verify there is only one header part.

A.1.5 Requirement 5

Test ID	**Conformance Test for Requirement 5** **http://www.opengis.net/spec/netcdf-binary/1.0/conf/common/fixed-size-data-part**
Test purpose	There shall be only one fixed-size data part per file
Test method	Open the dataset and verify that there is only one fixed-size data part

A.1.6 Requirement 6

Test ID	**Conformance Test for Requirement 6** **http://www.opengis.net/spec/netcdf-binary/1.0/conf/common/record-data-part**
Test purpose	There shall be only one record data part per file

Test method	Open the dataset and verify there is only one record data part.

A.1.7 Requirement 7

Test ID	**Conformance Test for Requirement 7** **http://www.opengis.net/spec/netcdf-binary/1.0/conf/common/BNF-for-header-non-recore-record**
Test purpose	The header, non-record and record parts shall conform to the BNF grammar segment given in 6.1.2.
Test method	Open the dataset and verify that the header, non-record and record parts shall conform to the BNF grammar segment given in 6.1.2.

A.1.8 Requirement 8

Test ID	**Conformance Test for Requirement 8** **http://www.opengis.net/spec/netcdf-binary/1.0/conf/common/header-part-specifications**
Test purpose	The header shall specify: • whether classic or 64-bit offset encoding (*magic)* is used • the length of the record dimension (*numrecs*) • the list of dimenisons (*dim_list*) • the list of global attributes (*gatt_list*) • the list of variables (*var_list*)
Test method	Open the dataset and verify that the header specifies: • whether classic or 64-bit offset encoding (*magic)* is used • the length of the record dimension (*numrecs*) • the list of dimenisons (*dim_list*)

	• the list of global attributes (*gatt_list*) • the list of variables (*var_list*)

A.1.9 Requirement 9

Test ID	**Conformance Test for Requirement 9** **http://www.opengis.net/spec/netcdf-binary/1.0/conf/common/BNF-for-header**
Test purpose	The header shall conform to the BNF grammar segment given in 6.1.2.2.
Test method	Open the dataset and verify that the header conforms to the BNF grammar segment given in 6.1.2.2.

A.1.10 Requirement 10

Test ID	**Conformance Test for Requirement 10** **http://www.opengis.net/spec/netcdf-binary/1.0/conf/common/contiguous-for-fixed-size-data**
Test purpose	The data for all non-record variables shall be stored contiguously for each variable, in the same order the variables occur in the header
Test method	Open the dataset and verify that data for all non-record variables is stored contiguously for each variable, in the same order the variables occur in the header.

.

A.1.11 Requirement 11

Test ID	**Conformance Test for Requirement 11** **http://www.opengis.net/spec/netcdf-binary/1.0/conf/common/block-values-for-fixed-size-data**
Test purpose	All data for a non-record variable shall be stored as a block of values of the same type as the variable, in row-major order (last dimension varying fastest).
Test method	Open the dataset and verify that all data for a non-record variable is stored as a block of values of the same type as the variable, in row-major order (last dimension varying fastest).

A.1.12 Requirement 12

Test ID	**Conformance Test for Requirement 12** **http://www.opengis.net/spec/netcdf-binary/1.0/conf/common/data-values-for non-record-variable**
Test purpose	The fixed-size data shall contain data values for variables that don't have an unlimited dimension, i.e., for each non-record variable.
Test method	Open the dataset and verify that the fixed-size data contain data values for variables that don't have an unlimited dimension, i.e., for each non-record variable.

A.1.13 Requirement 13

Test ID	**Conformance Test for Requirement 13** **http://www.opengis.net/spec/netcdf-binary/1.0/conf/common/data-row-major**
Test purpose	The data for each variable shall be stored contiguously, in row-major order for multi-dimensional variables.
Test	Open the dataset and verify that the data for each variable is stored

method	contiguously, in row-major order for multi-dimensional variables.

A.1.14 Requirement 14

Test ID	**Conformance Test for Requirement 14** **http://www.opengis.net/spec/netcdf-binary/1.0/conf/common/BNF-for-fixed-size-data**
Test purpose	The fixed-size (non-record) data shall conform to the BNF grammar segment given in 6.1.2.3.
Test method	Open the dataset and verify that the fixed-size (non-record) data conforms to the BNF grammar segment given in 6.1.2.3.

A.1.15 Requirement 15

Test ID	**Conformance Test for Requirement 15** **http://www.opengis.net/spec/netcdf-binary/1.0/conf/common/one-unlimited-dimension**
Test purpose	There shall be at most one unlimited dimension, the record dimension.
Test method	Open the dataset and verify that there is at most one unlimited dimension and that it is the record dimension.

A.1.16 Requirement 16

Test ID	**Conformance Test for Requirement 16** **http://www.opengis.net/spec/netcdf-binary/1.0/conf/common/data-values-for-unlimited-dimension**
Test purpose	The record data shall contain data values for variables that have an unlimited dimension.
Test	Open the dataset and verify that the record data contains data values for

method	variables that have an unlimited dimension.

A.1.17 Requirement 17

Test ID	**Conformance Test for Requirement 17** **http://www.opengis.net/spec/netcdf-binary/1.0/conf/common/current-size-in-header**
Test purpose	The current size of the record dimension shall be stored in the header which specifies how many records the file contains.
Test method	Open the dataset and verify that the current size of the record dimension is stored in the header and specifies how many records the file contains.

A.1.18 Requirement 18

Test ID	**Conformance Test for Requirement 18** **http://www.opengis.net/spec/netcdf-binary/1.0/conf/common/all-data-for-record-part**
Test purpose	Each record in the record data part shall contain all the data for that record for each record variable.
Test method	Open the dataset and verify that each record in the record data part contains all the data for that record for each record variable.

A.1.19 Requirement 19

Test ID	**Conformance Test for Requirement 19** **http://www.opengis.net/spec/netcdf-binary/1.0/conf/common/data-for-each-record**
Test purpose	Each record's worth of data for each record variable shall be stored contiguously, in row major order for multidimensional variables.

Test method	Open the dataset and verify that each record's worth of data for each record variable is stored contiguously, in row major order for multidimensional variables.

A.1.20 Requirement 20

Test ID	**Conformance Test for Requirement 20** **http://www.opengis.net/spec/netcdf-binary/1.0/conf/common/record-size**
Test purpose	All records shall be the same size, because they each contain all the data for a particular record for each record variable.
Test method	Open the dataset and verify that all records are the same size and that they each contain all the data for a particular record for each record variable.

A.1.21 Requirement 21

Test ID	**Conformance Test for Requirement 21** **http://www.opengis.net/spec/netcdf-binary/1.0/conf/common/BNF-for-record-data**
Test purpose	The record data section shall conform to the BNF grammar segment given in 6.1.2.4.
Test method	Open the dataset and verify that record data section conforms to the BNF grammar segment given in 6.1.2.4.

A.1.22 Requirement 22

Test ID	**Conformance Test for Requirement 22** **http://www.opengis.net/spec/netcdf-binary/1.0/conf/common/BNF-definitions**

Test purpose	The BNF segments in the previous requirements shall conform to the BNF specifications in 6.1.2.5.
Test method	Open the dataset and verify that BNF segments in the previous requirements conform to the BNF specifications in 6.1.2.5.

A.2 Test Class: netCDF Binary Classic Format

Applies for conformance class netCDF binary with URI
http://www.opengis.net/spec/netcdf-binary/1.0/conf/classic

A.2.1 Requirement 23

Test ID	**Conformance Test for Requirement 23** **http://www.opengis.net/spec/netcdf-binary/1.0/conf/classic/Version-Offset**
Test purpose	A netCDF Classic Variant dataset shall conform to all the requirements of the netCDF Classic encoding with the VERSION and OFFSET specifications given in 6.1.3.
Test method	Open the netCDF Classic Variant dataset and verify that it conforms to all the requirements of the netCDF Classic encoding with the VERSION and OFFSET specifications given in 6.1.3.

A.3 Test Class: netCDF Binary 64-bit Offset Format

Applies for conformance class netCDF binary with URI
http://www.opengis.net/spec/netcdf-binary/1.0/conf/64-bit-offset

A.3.1 Requirement 24

Test ID	**Conformance Test for Requirement 24** **http://www.opengis.net/spec/netcdf-binary/1.0/conf/64-bit-offset/Version-Offset**
Test purpose	A netCDF 64-bit Offset Variant dataset shall conform to all the requirements of the netCDF Classic encoding with the VERSION and OFFSET specifications given in 6.1.4.

Test method	Open the netCDF 64-bit Offset Variant dataset and verify that it conforms to all the requirements of the netCDF Classic encoding with the VERSION and OFFSET specifications given in 6.1.4.

Annex B: Complete BNF Grammar (Normative)

B.1 Complete BNF Grammar for netCDF Classic and 64-bit Offset Binary Encoding

Note that this BNF grammar and the portions of it in the document body are verbatim quotes from

NASA ESDS-RFC-011v2.00 R. Rew, E. Hartnett, D. Heimbigner, E. Davis, J. Caron: *NetCDF Classic and 64-bit Offset File Formats*

http://www.esdswg.org/spg/rfc/esds-rfc-011/ESDS-RFC-011v2.00.pdf

To present the format more formally, we use a BNF grammar notation. In this notation:

- Non-terminals (entities defined by grammar rules) are in lower case.
- Terminals (atomic entities in terms of which the format specification is written) are in upper case, and are specified literally as US-ASCII characters within single-quote characters or are described with text between angle brackets (‘<’ and ‘>’).
- Optional entities are enclosed between braces (‘[’ and ‘]’).
- A sequence of zero or more occurrences of an entity is denoted by ‘[entity ...]’.
- A vertical line character (‘|’) separates alternatives. Alternation has lower precedence than concatenation.
- Comments follow ‘//’ characters.
- A single byte that is not a printable character is denoted using a hexadecimal number with the notation ‘\xDD’, where each D is a hexadecimal digit.
- A literal single-quote character is denoted by ‘\’’, and a literal back-slash character is denoted by ‘\\’.

Following the grammar, a few additional notes are included to specify format characteristics that are impractical to capture in a BNF grammar, and to note some special cases for implementers. Comments in the grammar point to the notes and special cases, and help to clarify the intent of elements of the format.

```
netcdf_file  = header data
header       = magic numrecs dim_list gatt_list var_list
magic        = 'C' 'D' 'F' VERSION
VERSION      = \x01 |                 // classic format
               \x02                   // 64-bit offset format
numrecs      = NON_NEG | STREAMING    // length of record dimension
dim_list     = ABSENT | NC_DIMENSION nelems [dim ...]
gatt_list    = att_list               // global attributes
```

```
att_list     = ABSENT | NC_ATTRIBUTE nelems [attr ...]
var_list     = ABSENT | NC_VARIABLE nelems [var ...]
ABSENT       = ZERO ZERO               // Means list is not present
ZERO         = \x00 \x00 \x00 \x00      // 32-bit zero
NC_DIMENSION = \x00 \x00 \x00 \x0A      // tag for list of dimensions
NC_VARIABLE  = \x00 \x00 \x00 \x0B      // tag for list of variables
NC_ATTRIBUTE = \x00 \x00 \x00 \x0C      // tag for list of attributes
nelems       = NON_NEG              // number of elements in following sequence
dim          = name dim_length
name         = nelems namestring
                    // Names a dimension, variable, or attribute.
                   // Names should match the regular expression
                   //([a-zA-Z0-9_]|{MUTF8})([^\x00-\x1F/\x7F-\xFF]|{MUTF8})*
                   // For other constraints, see “Note on names”, below.
namestring   = ID1 [IDN ...] padding
ID1          = alphanumeric | '_'
IDN          = alphanumeric | special1 | special2
alphanumeric = lowercase | uppercase | numeric | MUTF8
lowercase    = 'a'|'b'|'c'|'d'|'e'|'f'|'g'|'h'|'i'|'j'|'k'|'l'|'m'|
               'n'|'o'|'p'|'q'|'r'|'s'|'t'|'u'|'v'|'w'|'x'|'y'|'z'
uppercase    = 'A'|'B'|'C'|'D'|'E'|'F'|'G'|'H'|'I'|'J'|'K'|'L'|'M'|
               'N'|'O'|'P'|'Q'|'R'|'S'|'T'|'U'|'V'|'W'|'X'|'Y'|'Z'
numeric      = '0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9'
                    // special1 chars have traditionally been
                    // permitted in netCDF names.
special1     = '_'|'.'|'@'|'+'|'-'
                    // special2 chars are recently permitted in
                    // names (and require escaping in CDL).
                    // Note: '/' is not permitted.
special2     = ' ' | '!' | '"' | '#'  | '$' | '%' | '&' | '\'' |
               '(' | ')' | '*' | ','  | ':' | ';' | '<' | '='  |
               '>' | '?' | '[' | '\\' | ']' | '^' | '`' | '{'  |
               '|' | '}' | '~'
MUTF8        = <multibyte UTF-8 encoded, NFC-normalized Unicode character>
dim_length   = NON_NEG       // If zero, this is the record dimension.
                             // There can be at most one record dimension.
attr         = name nc_type nelems [values ...]
nc_type      = NC_BYTE | NC_CHAR | NC_SHORT | NC_INT | NC_FLOAT | NC_DOUBLE
var          = name nelems [dimid ...] vatt_list nc_type vsize begin
                             // nelems is the dimensionality (rank) of the
                             // variable: 0 for scalar, 1 for vector, 2
                             // for matrix, ...
dimid        = NON_NEG       // Dimension ID (index into dim_list) for
                             // variable shape. We say this is a "record
                             // variable" if and only if the first
                             // dimension is the record dimension.
vatt_list    = att_list      // Variable-specific attributes
vsize        = NON_NEG       // Variable size. If not a record variable,
                             // the amount of space in bytes allocated to
                             // the variable's data. If a record variable,
                             // the amount of space per record. See “Note on
                             // vsize” below.
begin        = OFFSET        // Variable start location. The offset in
                             // bytes (seek index) in the file of the
                             // beginning of data for this variable.
data         = non_recs recs
non_recs     = [vardata ...] // The data for all non-record variables,
                             // stored contiguously for each variable, in
                             // the same order the variables occur in the
                             // header.
vardata      = [values ...]  // All data for a non-record variable, as a
                             // block of values of the same type as the
                             // variable, in row-major order (last
                             // dimension varying fastest).
recs         = [record ...]  // The data for all record variables are
                             // stored interleaved at the end of the
                             // file.
record       = [varslab ...] // Each record consists of the n-th slab
                             // from each record variable, for example
                             // x[n,...], y[n,...], z[n,...] where the
                             // first index is the record number, which
                             // is the unlimited dimension index.
varslab      = [values ...]  // One record of data for a variable, a
                             // block of values all of the same type as
                             // the variable in row-major order (last
                             // index varying fastest).
values       = bytes | chars | shorts | ints | floats | doubles
string       = nelems [chars]
bytes        = [BYTE ...] padding
chars        = [CHAR ...] padding
shorts       = [SHORT ...] padding
ints         = [INT ...]
floats       = [FLOAT ...]
doubles      = [DOUBLE ...]
padding      = <0, 1, 2, or 3 bytes to next 4-byte boundary>
                             // Header padding uses null (\x00) bytes. In
                             // data, padding uses variable's fill value.
                             // See “Note on padding” below for a special
                             // case.
NON_NEG      = <non-negative INT>
STREAMING    = \xFF \xFF \xFF \xFF   // Indicates indeterminate record
                                     // count, allows streaming data
OFFSET       = <non-negative INT> |  // For classic format or
               <non-negative INT64>  // for 64-bit offset format
BYTE         = <8-bit byte>          // See “Note on byte data”, below.
CHAR         = <8-bit byte>          // See “Note on char data”, below.
SHORT        = <16-bit signed integer, Bigendian, two's complement>
INT          = <32-bit signed integer, Bigendian, two's complement>
INT64        = <64-bit signed integer, Bigendian, two's complement>
FLOAT        = <32-bit IEEE single-precision float, Bigendian>
DOUBLE       = <64-bit IEEE double-precision float, Bigendian>
                             // following type tags are 32-bit integers
NC_BYTE      = \x00 \x00 \x00 \x01       // 8-bit signed integers
NC_CHAR      = \x00 \x00 \x00 \x02       // text characters
NC_SHORT     = \x00 \x00 \x00 \x03       // 16-bit signed integers
NC_INT       = \x00 \x00 \x00 \x04       // 32-bit signed integers
NC_FLOAT     = \x00 \x00 \x00 \x05       // IEEE single precision floats
NC_DOUBLE    = \x00 \x00 \x00 \x06       // IEEE double precision floats
                           // Default fill values for each type, may be
                           // overridden by variable attribute named
                           // ‘_FillValue’, see “Note on fill values”, below
FILL_BYTE    = \x81                       // (signed char) -127
FILL_CHAR    = \x00                       // null byte
FILL_SHORT   = \x80 \x01                  // (short) -32767
FILL_INT     = \x80 \x00 \x00 \x01        // (int) -2147483647
FILL_FLOAT   = \x7C \xF0 \x00 \x00           // (float) 9.9692099683868690e+36
FILL_DOUBLE  = \x47 \x9E \x00 \x00 \x00 \x00 // (double)9.9692099683868690e+36
```

Note on vsize: This number is the product of the dimension lengths (omitting the record dimension) and the number of bytes per value (determined from the type), increased to the next multiple of 4, for each variable. If a record variable, this is the amount of space per record. The netCDF “record size” is calculated as the sum of the vsize's of all the record variables.

The vsize field is actually redundant, because its value may be computed from other information in the header. The 32-bit vsize field is not large enough to contain the size of variables that require more than 2^{32} - 4 bytes, so 2^{32} - 1 is used in the vsize field for such variables.

Note on names: Earlier versions of the netCDF C-library reference implementation enforced a more restricted set of characters in creating new names, but permitted reading names containing arbitrary bytes. This RFC extends the permitted characters in names to include multi-byte UTF-8 encoded[7] Unicode[4] and additional printing characters from the US-ASCII alphabet. The first character of a name must be alphanumeric, a multi-byte UTF-8 character, or '_' (traditionally reserved for names with meaning to implementations, such as the “_FillValue” attribute). Subsequent characters may also include printing special characters, except for '/' which is not allowed in names. Names that have trailing space characters are also not permitted.

Implementations of the netCDF classic and 64-bit offset format must ensure that names are normalized according to Unicode NFC normalization rules [5] during encoding as UTF-8 for storing in the file header. This is necessary to ensure that gratuitous differences in the representation of Unicode names do not cause anomalies in comparing files and querying data objects by name.

Note on streaming data: The largest possible record count, 2^{32}-1, is reserved to indicate an indeterminate number of records. This means that the number of records in the file must be determined by other means, such as reading them or computing the current number of records from the file length and other information in the header. It also means that the numrecs□field in the header will not be updated as records are added to the file.

Note on padding: In the special case of only a single record variable of character, byte, or short type, no padding is used between data values.

Note on byte data: It is possible to interpret byte data as either signed (-128 to 127) or unsigned (0 to 255). When reading byte data through an interface that converts it into another numeric type, the default interpretation is signed. There are various attribute conventions for specifying whether bytes represent signed or unsigned data, but no standard convention has been established. The variable attribute “_Unsigned” is reserved for this purpose in future implementations.

Note on char data: Although the characters used in netCDF names must be encoded as UTF-8, character data may use other encodings. The variable attribute “_Encoding” is reserved for this purpose in future implementations.

Note on fill values: Because data variables may be created before their values are written, and because values need not be written sequentially in a netCDF file, default “fill values” are defined for each type, for initializing data values before they are explicitly written. This makes it possible to detect reading values that were never written. The variable attribute "_FillValue", if present, overrides the default fill value for a variable. If _FillValue is defined then it should be scalar and of the same type as the variable.

Fill values are not required, however, because netCDF libraries have traditionally supported a "no fill" mode when writing, omitting the initialization of variable values with fill values. This makes the creation of large files faster, but also eliminates the possibility of detecting the inadvertent reading of values that were not written.

Annex C: Revision history

Date	Release	Author	Paragraph modified	Description
2010-08-27	1.0.0	Ben Domenico	All	Created
2010-12-28	1.0.1	Ben Domenico	6, Annex A	Added leading "." for relative URIs. Changed "req" to "conf" in conformance class URIs.
2010-12-28	1.0.1	Ben Domenico	2, 3, 5,Annex A	Changed conformance class URIs according to recommendation of OGC Naming Authority
2010-01-07	1.0.1	Ben Domenico	5.3	Removed Table 2 because there are no XML schemas requiring namespace mappings.
2010-01-07	1.0.1	Ben Domenico	Table of Contents	Updated Table of Contents for page number changes due to editing.
2011-01-07	1.0.1	Ben Domenico	All	Change document release number r2 to r3
2011-02-16	1.0.1	Ben Domenico	Various	Prepare for publication