netCDF-4/HDF5 File Format

1   Status of this Memo

This is a description of a NASA Earth Science Data Systems (ESDS) Community Standard.

Distribution of this memo is unlimited.

2   Change Explanation

V0.01 – Initial version.

V0.02 – Removed section on generating netCDF-4 files with the HDF5 library as a separate technical note. Incorporated other changes suggested by NASA review committee.

V0.03 – Restored section on generating netCDF-4 files with the HDF5 library as Appendix B.

3   Copyright Notice

4   Abstract

This document nominates the netCDF-4/HDF5 File Format for adoption as a NASA ESDS community standard. It specifies the netCDF-4/HDF5 file format independent of the netCDF I/O libraries designed to read and write netCDF-4/HDF5 data. The netCDF-4/HDF5 file format enables the expansion of the netCDF model, libraries, and machine-independent data format for geoscience data. Together the netCDF interfaces, libraries, and formats support the creation, access, and sharing of scientific data.

With suitable community conventions, the netCDF-4/HDF5 data format can help improve the capability to read and share important scientific data among data providers, data users, and data services.

5   Table of Contents

6   Introduction

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

NetCDF enables the creation of collections of data that are:

- Self-Describing: NetCDF datasets include information about the data they contain.

- Portable: The same netCDF data file can be used on computers with different means of storing integers, characters, and floating-point numbers.

- Direct-access: A small subset of a large data set may be accessed efficiently, without first reading through all the preceding data.

- Appendable: Data may be appended to a properly structured netCDF file without copying the data set or redefining its structure.

- Sharable: One writer and multiple readers may simultaneously access the same netCDF file. Using parallel netCDF interfaces, multiple writers may write a file concurrently.

- Archivable: Access to current and earlier forms of netCDF data will be supported by current and future versions of the software.

ESDS-RFC-022v1   **Edward Hartnett**
Category: Standards Track   **March 2011**
Updates/Obsoletes: None   **netCDF-4/HDF5 File Format**

In different contexts, "netCDF" may refer to an abstract data model, a software implementation with associated application program interfaces (APIs), or a data format. Confusion may easily arise in discussions of different versions of the data models, software, and formats, because the relationships among versions of these entities are more complex than a simple one-to-one correspondence. For example, compatibility commitments require that new versions of the software support all previous variants of the format and data model.

To avoid this potential confusion, we assign distinct names to versions of the formats, data models, and software releases that will be used consistently in the remainder of this document.

This document formally specifies two format variants for netCDF data, the netCDF-4/HDF5 format and the netCDF-4/HDF5 classic model format. The other two netCDF formats, classic format and 64-bit offset format, are documented in ESDS-RFC-011. NetCDF Classic and 64-bit Offset File Formats [2], and are outside the scope of this document. The classic format was the only available format for netCDF data from 1989 through 2004. In 2004, the 64-bit offset format variant was introduced for creation of and access to larger files. The reference software, available for C-based and Java-based programs, supported the use of the same APIs for accessing either classic or 64-bit offset files. Thus, software programs that read netCDF files would not have to depend on which format was used. The netCDF-4/HDF5 and netCDF-4/HDF5 classic model formats were introduced with netCDF-4.0 in 2008.

There are only two netCDF data models, the classic model and the enhanced model. The classic model is the simpler of the two, and is used for all data stored in classic format, 64-bit offset format, or netCDF-4 classic model format. The enhanced model (also referred to as the netCDF-4 data model) is an extension of the classic model. The enhanced model incorporates more powerful forms of data representation and data types at the expense of some additional complexity. Data represented with the classic model can also be represented using the enhanced model. However, datasets that use features of the enhanced model, such as user-defined nested data types, cannot be represented with the classic model. Use of added features of the enhanced model requires that data be stored in the netCDF-4/HDF5 format.

Versions 1.0 through 3.5 of the Unidata C-based reference software, released between 1989 and 2000, supported only the classic data model and classic format. Version 3.6, released in late 2004, provided support for the 64-bit offset format, but still used the classic data model. With Version 4.0, released in 2008, Unidata introduced the enhanced data model along with the two new HDF5-based [1] format variants, the netCDF-4/HDF5 format and the netCDF-4/HDF5 classic model format. Regardless of the ongoing evolution of the data models, formats, and APIs, Unidata is committed to continue to support all previous netCDF data models, data format variants, and APIs in future software releases.

Use of the HDF5 storage layer in netCDF-4 software provides features for improved performance, independent of the data model used. These new features include compression and dynamic schema changes. Such performance improvements are available for data stored in the netCDF-4/HDF5 classic model format, even when accessed by programs that only support the classic model. Related formats not discussed in this document include CDL ("Common Data

Language", the original ASCII form of binary netCDF data), and NcML (NetCDF Markup Language, an XML-based representation for netCDF metadata and data).

Knowledge of format details is not required to read or write netCDF datasets. Software that reads netCDF data using the reference implementation automatically detects and uses the correct version of the format for accessing data. On the other hand, some understanding of these details may be helpful for assessing performance issues related to disk or server access.

The netCDF reference library, developed and supported by Unidata, is written in C, with Fortran77, Fortran90, and C++ interfaces. A number of community and commercially supported interfaces to other languages are also available, including IDL, Matlab, Perl, Python, and Ruby.

An independent implementation, also developed and supported by Unidata, is written entirely in Java. It can read, but not yet write, netCDF-4/HDF5 files.

6.1   Motivation for this RFC

No specification that describes the format in sufficient detail for independent implementations of netCDF/HDF5 access software is currently available. The establishment of an ESDS standard requires a reference that precisely documents the netCDF-4/HDF5 format. A published reference standard also assures the long-term usability of netCDF-4/HDF5 data archives, as it transcends issues concerning the future availability of specific hardware and software.

As current and potential developers learn of the simplicity and representational power of a widely used format, the existence of this documented standard should encourage increased interoperability of data services, scientific analysis software, and data management software.

Because netCDF-4 implements the netCDF classic and enhanced data models using HDF5 as the storage layer, a complete specification of the netCDF-4/HDF5 file formats would necessarily incorporate the HDF5 file specification. A complete specification of HDF5 files is provided in ESDS-RFC-007 "HDF5 Data Model, File Format and Library – HDF5 1.6" [1]. For more information about HDF5, see: http://www.hdfgroup.org/HDF5/ [4].

The netCDF classic file format is specified in ESDS-RFC-011 "NetCDF Classic and 64-bit Offset File Formats" [2]. This document, therefore, describes the new features available with netCDF-4, and the additional data types and structures in the netCDF-4 enhanced data model.

Although every file in netCDF-4 format is an HDF5 file, HDF5 files are not necessarily netCDF-4 format files, because the netCDF-4 format intentionally uses a limited subset of the HDF5 data model and file format features. All HDF5 features and capabilities incorporated into netCDF4 are described in sections 7.3 and 7.4. Appendix B of this document provides guidance on creating valid netCDF-4 data files using the HDF5 library and API.

A wealth of documentation, including the "NetCDF Users' Guide" [5] is available online at http://www.unidata.ucar.edu/software/netcdf/ [6].

7 NetCDF-4 Use and Features

Using HDF5 as the underlying storage layer, netCDF-4 files remove many of the restrictions for classic and 64-bit offset files. The richer enhanced model supports user-defined types and data structures, hierarchical scoping of names using groups as well as additional primitive types including strings, larger variable sizes, and multiple unlimited dimensions. The underlying HDF5 storage layer also supports per-variable compression, multidimensional tiling, and efficient dynamic schema changes, so that data need not be copied when adding new variables to the file schema.

7.1 Binary Formats

When netCDF was first introduced (over 20 years ago), there was only one output format, which we now refer to as "netCDF classic format."

The classic format contains size limitations, which created problems for some users. Thus, Unidata released a new binary format in version 3.6.0 in 2004, which used 64-bit offsets, instead of the 32-bit offsets. This binary format is called the "64-bit offset format."

With the release of netCDF-4 (in 2008) another binary format was introduced, which uses HDF5. This format is called the netCDF-4/HDF5 format.

The original, classic format is still (and will always remain) the default netCDF format. The two additional formats, 64-bit offset and netCDF-4/HDF5 formats, are available with the use of an additional flag when creating the file.

It is not necessary for the user to know what type of binary format is being used. The netCDF library will determine the binary format automatically when the file is opened. However, older versions of netCDF can only understand the binary formats available at the time of their release. If the user has a 3.6.1 version of netCDF installed, the software will not be able to open files in the netCDF-4/HDF5 format. Similarly, if the user has a 3.5.1 version of the library installed, the software will not be able to understand any format other than the original classic format.

For this reason, and to take advantage of bug fixes and performance improvements, users are strongly advised to update old netCDF installations to the latest version.

7.2 The Classic and Enhanced Data Models

The following diagram shows the enhanced netCDF data model, which is a superset of the classic netCDF data model. Elements of the classic model are shown in black, elements that exist only in the enhanced model are shown in red.

The enhanced data model offers much more expressive power for complex data sets. Using user-defined types, groups, multiple unlimited dimensions, and the full range of primitive types, enable creation of extremely complex data sets.
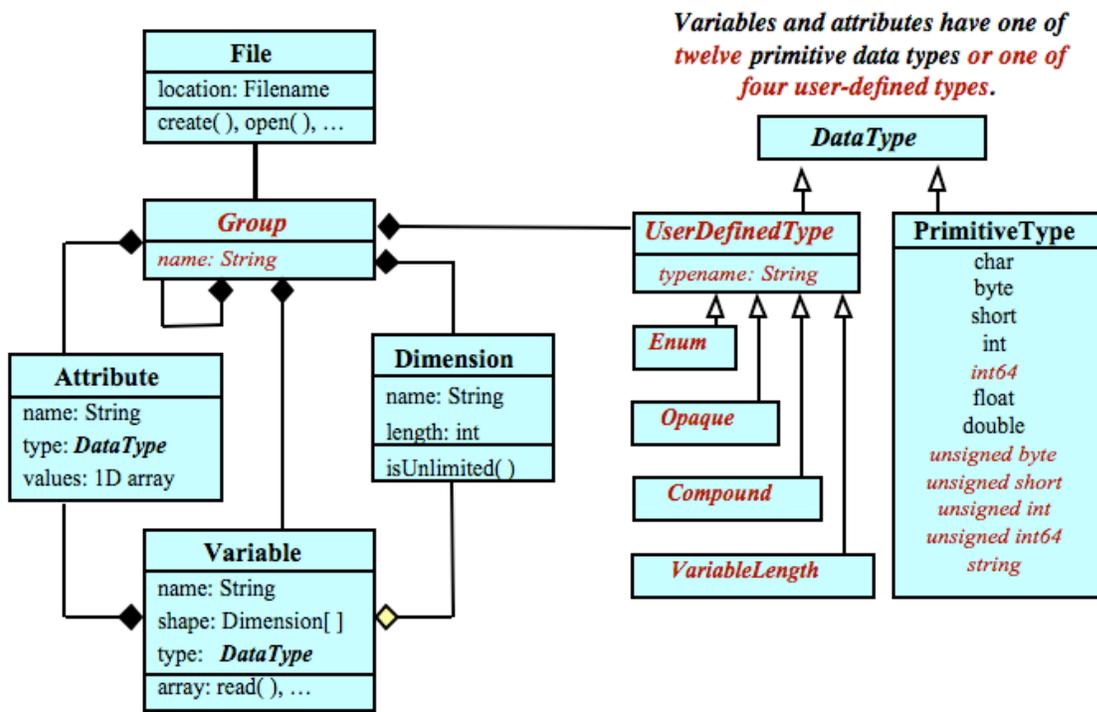
Variables and attributes have one of twelve primitive data types or one of four user-defined types.

A file has a top-level unnamed group. Each group may contain one or more named subgroups, user-defined types, variables, dimensions, and attributes. Variables also have attributes. Variables may share dimensions, indicating a common grid. One or more dimensions may be of unlimited length.

This expressive power comes at a cost: most existing netCDF software handles only the classic data model. Data sets that use features from the enhanced model will require the use of software that can handle the additional complexity.

Users are advised to use the classic data model wherever it works well. Users should restrict the use of the enhanced model to those data sets that cannot easily be expressed in the classic model.

All files that appear in the classic netCDF format must use only the classic model. It is possible and often useful, to create classic model files in netCDF-4/HDF5 format. Some of the features of netCDF-4/HDF5 files do not rely on the enhanced model. These features include compression, parallel I/O, relaxed size limits, and the performance benefits of chunking and endianness control. These features may be used while still maintaining full code compatibility with existing netCDF software.

7.3   Using NetCDF-4/HDF5 Files with the Classic Model

The classic model consists of variables, dimensions, and attributes. The classic model employs six data types. NetCDF-4.0 introduced the enhanced model, but users who stay within the

ESDS-RFC-022v1   **Edward Hartnett**
Category: Standards Track   **March 2011**
Updates/Obsoletes: None   **netCDF-4/HDF5 File Format**

boundaries of the classic model will reap the very significant benefit of backwards software compatibility with netCDF-4/HDF5 data files. An existing software package may be recompiled with the latest netCDF release, and, without any code changes, will be able to read netCDF-4/HDF5 files which conform to the classic format. Only the "file create" command needs to be changed for code that writes data.

An example will help illustrate this feature. The Global Modeling and Assimilation Office (GMAO) at the Goddard Space Flight Center outputs assimilated data in the netCDF format, and has for many years. Users of the data have many custom tools and scripts which read the data files. The GMAO data files were becoming very large – too large for the classic model. Additionally, the very large data files took a lot of time to transfer around the network. To address this, the GMAO programmers changed their code to create netCDF-4/HDF5 files, and subsequently turned on compression for the data variables. The result is a netCDF-4/HDF5 file which is significantly smaller and thus does not have the variable size limits of the classic format file.

Users did not have to change their code to handle this new output format. They recompiled their tools with a new release of the netCDF library, and those tools continue to work just as before. The recompiled code can transparently read the compressed netCDF-4/HDF5 files, in addition to any classic format files.

7.3.1   Size Limits

The following table shows the limits which apply to the three binary formats of netCDF.

Limit	Classic Format	64-bit Offset Format	NetCDF-4/HDF5 Format
Max File Size	8 EiB	8 EiB	Any size supported by file system.
Max Number of Fixed Vars > 2 GiB	1 (last)	2^32	No limit.
Max Record Vars w/ Rec Size > 2 GiB	1 (last)	2^32	No limit.
Max Size of Fixed/Record Size of Record Var	2 GiB	4 GiB	No limit.
Max Record Size	8 EiB/nrecs	8 EiB/nrecs	No limit.

7.3.2   Compression

Compressed data is transparently uncompressed when read by the HDF5 library. Data producers can turn on data compression, and data users will not need to make any changes in their code to read the compressed data.   The compression library used is gzip.   Users can estimate the compressibility of a data file by employing the gzip utility on the UNIX/Linux command line.

7.3.3   Chunking and Endianness Control

Data in a netCDF-4/HDF5 file are written in "chunks". The size of each chunk may be adjusted by the data producer. Data producers may also control the endianness of data for netCDF-4/HDF5 files. (Classic and 64-bit offset files are always in big-endian format.)

Data producers can set the chunk sizes and endianness of netCDF-4/HDF5 data variables to significantly increase read or write performance. No changes need be made in reading code to achieve these performance gains.

7.3.4   Parallel I/O

It is possible to use parallel I/O when reading or writing netCDF-4/HDF5 files. Using parallel I/O can significantly enhance computational performance. Files written or read using parallel I/O are netCDF-4/HDF5 files, and may also be written/read by sequential applications.

7.3.5   Classic Model Flag

The netCDF-4 library provides a CLASSIC_MODEL flag.   Incorporation of the CLASSIC MODEL flag at the time of creation of a netCDF-4/HDF5 file enforces the limitations of the classic model on that file.   By using the classic model flag, data producers can be sure that they are generating classic model data, that will be readable by any existing classic model netCDF software.

7.4   Using NetCDF-4/HDF5 files with the Enhanced Model

The enhanced data model provides additional ways to organize complex data sets.

7.4.1   Groups

Groups can be used to organize data hierarchically. Conceptually, each group functions as an entire data file in the classic netCDF model, with dimensions, variables, and attributes global to the group. At the very least, every netCDF-4/HDF5 file contains one, unnamed, root group.

Use of groups in data sets will require that read software be written to make use of the group structure.

7.4.2   New Data Types

NetCDF-4/HDF5 data format adds the following primitive types: byte, unsigned byte, unsigned short, unsigned int, signed 64-bit int, unsigned 64-bit int, and string.

ESDS-RFC-022v1   **Edward Hartnett**
Category: Standards Track   **March 2011**
Updates/Obsoletes: None   **netCDF-4/HDF5 File Format**

Additionally, users can define new data types. To define these types, the data producer must call the appropriate type definition functions, and provide the data to the netCDF library in accordance with the specified layout of the new data type. The following categories of new types can be created:

- ⚔ Compound types – like C structure

- ⚔ Variable length types (VLENs) – arrays with variable length

- ⚔ Enum types – like C enumerated type

- ⚔ Opaque types – opaque bags of bits of a fixed size

The use of user-defined types permits greater expressiveness in data output, but at the cost of greater complexity for reading software.

7.4.3 Multiple Unlimited Dimensions

In the classic and 64-bit offset format, every array dimension except one must have a fixed length. The one extensible dimension is known as the unlimited dimension. In netCDF-4/HDF5 files, this restriction is removed. Any number of unlimited dimensions may be used.

7.5 NetCDF-4 and HDF5 Interoperability

A netCDF-4/HDF5 file created with netCDF-4 software is a legitimate HDF5 file. The features of netCDF-4 are a subset of the features of HDF5, so the resulting file can be used by any existing HDF5 application.

Although every file in netCDF-4 format is an HDF5 file, HDF5 files are not necessarily netCDF-4 format files, because the netCDF-4 format intentionally uses a limited subset of the HDF5 data model and file format features, as described in sections 7.3 and 7.4 of this document. Some HDF5 features not supported in the netCDF enhanced model and netCDF-4 format include non-hierarchical group structures, HDF5 reference types, multiple links to a data object, user-defined atomic data types, stored property lists, more permissive rules for data object names, the HDF5 date/time type, and attributes associated with user-defined types.

See Appendix B for information on creating a valid netCDF-4 file using the HDF5 software library and API.

8 References

Normative References

[1] ESDS-RFC-007 HDF5 Data Model, File Format and Library – HDF5 1.6

[2] ESDS-RFC-011 NetCDF Classic and 64-bit Offset File Formats

Informative References

[3] ESDS-RFC-021 NetCDF Climate and Forecast (CF) Metadata Conventions

[4] HDF5 web site, http://www.hdfgroup.org/HDF5/

[5] NetCDF User's Guide, http://www.unidata.ucar.edu/software/netcdf/docs/netcdf.html

[6] NetCDF web site, http://www.unidata.ucar.edu/software/netcdf/

9   Authors' Address

Ed Hartnett, UCAR Unidata, P.O. Box 3000, Boulder CO 80307-3000, USA, email: edh@ucar.edu

Appendix A - Acronyms

Acronym   Description

API: application programming interface

ASCII: American Standard Code for Information Interchange

CDL: Common Data Language

ESDS: Earth Science Data Systems

GMAO: Global Modeling and Assimilation Office

HDF5:   Hierarchical Data Format, version 5

IDL: Interactive Data Language

NASA: National Aeronautics and Space Administration

NcML: netCDF Markup Language

netCDF: network Common Data Form

I/O: input / output

RFC: Request for Comment

UCAR: University Corporation for Atmospheric Research

VLEN: Variable length types

XML: eXtensible Markup Language

Appendix B – Creating a valid netCDF4 file using the HDF5 API

This Appendix provides guidelines that will enable HDF5 users to create files that will be accessible from netCDF-4.

B-1   Creation Order

The netCDF API maintains the order of objects as they were created in the file. The same is not true in HDF5, which maintains the objects in alphabetical order. Starting in version 1.8 of HDF5, the ability to maintain creation order was added. To enable netCDF4 compliance, this must capability be explicitly turned on in the HDF5 data file.

Each group must have a link and attribute creation order set. The following code (from libsrc4/nc4hdf.c) shows how the netCDF-4 library sets these when creating a group.

```
/* Create group, with link_creation_order set in the group
 * creation property list. */

if ((gcpl_id = H5Pcreate(H5P_GROUP_CREATE)) < 0)
    return NC_EHDFERR;

if (H5Pset_link_creation_order (gcpl_id, H5P_CRT_ORDER_TRACKED |
    H5P_CRT_ORDER_INDEXED) < 0)
    BAIL(NC_EHDFERR);

if (H5Pset_attr_creation_order(gcpl_id, H5P_CRT_ORDER_TRACKED |
    H5P_CRT_ORDER_INDEXED) < 0)
    BAIL(NC_EHDFERR);

if ((grp->hdf_grpid = H5Gcreate2(grp->parent->hdf_grpid, grp->name, H5P_DEFAULT,
    gcpl_id, H5P_DEFAULT)) < 0)
    BAIL(NC_EHDFERR);

if (H5Pclose(gcpl_id) < 0)
    BAIL(NC_EHDFERR);
```

Each dataset in the HDF5 file must be created with a property list that includes the creation order attribute set for creation ordering. The H5Pset_attr_creation_order function is used to set the creation ordering for the attributes of a given variable. The following example code (from libsrc4/nc4hdf.c) shows how the creation ordering is turned on by the netCDF library.

```
/* Turn on creation order tracking. */

if (H5Pset_attr_creation_order(plistid, H5P_CRT_ORDER_TRACKED |
    H5P_CRT_ORDER_INDEXED) < 0)
    BAIL(NC_ EHDFERR);
```

ESDS-RFC-022v1   **Edward Hartnett**
Category: Standards Track   **March 2011**
Updates/Obsoletes: None   **netCDF-4/HDF5 File Format**

B-2   Groups

NetCDF-4 groups are the same as HDF5 groups, but groups in a netCDF-4 file must be strictly hierarchical. In general, HDF5 permits non-hierarchical structuring of groups (for example, a group that is its own grandparent). These non-hierarchical relationships are not allowed in netCDF-4 files.

In the netCDF API, the global attribute becomes a group-level attribute. That is, each group may have its own global attributes.

The root group of a file is named "/" in the netCDF API, where names of groups are used. The netCDF API (like the HDF5 API) makes little use of names, and refers to entities by number.

B-3   Dimensions with HDF5 Dimension Scales

Until version 1.8, HDF5 did not have any capability to represent shared dimensions. With the 1.8 release, HDF5 introduced the dimension scale feature to allow shared dimensions in HDF5 files.

The HDF5 dimension scale is not exactly equivalent to the netCDF shared dimension. This leads to a number of compromises in the design of netCDF-4.

A netCDF shared dimension consists solely of a length and a name. An HDF5 dimension scale also includes values for each point along the dimension. This additional information is (optionally) included in a netCDF coordinate variable.

To handle the case of a netCDF dimension without a coordinate variable, netCDF-4 creates dimension scales of type char, and leaves the contents of the dimension scale empty. Only the name and length of the scale are significant. To distinguish this case, netCDF-4 takes advantage of the NAME attribute of the dimension scale. In the case of dimensions without coordinate data, the HDF5 dimension scale NAME attribute is set to the string: "This is a netCDF dimension but not a netCDF variable." The NAME attribute of the scale should not be confused with the name of the scale itself.

In the case where a coordinate variable is defined for a dimension, the HDF5 dimscale matches the type of the netCDF coordinate variable, and contains the coordinate data.

A further difficulty arises when an n-dimensional coordinate variable is defined, where n is greater than one. NetCDF allows such coordinate variables, but the HDF5 model does not allow dimension scales to be attached to other dimension scales, making it impossible to completely represent the multi-dimensional coordinate variables of the netCDF model.

To capture this information, multidimensional coordinate variables have an attribute named _Netcdf4Coordinates. The attribute is an array of H5T_NATIVE_INT, with the netCDF dimension IDs of each of its dimensions.

The _Netcdf4Coordinates attribute is otherwise hidden by the netCDF API. It does not appear as one of the attributes for the netCDF variable involved, except through the HDF5 API.

B-4   Dimensions without HDF5 Dimension Scales

Starting with the netCDF-4.1 release, netCDF can read HDF5 files which do not use dimension scales. The netCDF library assigns dimensions to the HDF5 dataset as needed, based on the length of the dimension.

When an HDF5 file is opened, the netCDF API examines each dataset in the file. The API determines the lengths of all the dimensions in the shape of the dataset. Each new (i.e. previously unencountered) length results in the creation of a phony dimension in the netCDF API.

This logic will not accurately detect a shared, unlimited dimension in the HDF5 file when different datasets have different lengths along this dimension (possible in HDF5, but not in netCDF).

Note that this is a read-only capability for the netCDF library. When the netCDF library writes HDF5 files, the netCDF software always applies a dimension scale for every dimension.
Datasets must have either dimension scales for every dimension, or no dimension scales at all. Partial dimension scales are not, at this time, understood by the netCDF library.

B-5   Dimension and Coordinate Variable Ordering

The netCDF-4 library writes variables in their creation order. Since some variables are also dimension scales, their order reflects both the order of the dimensions and the order of the coordinate variables.

The order of dimension and the order of the coordinate variables may be different. Consider the following code:

```
/* Create a test file. */
if (nc_create(FILE_NAME, NC_CLASSIC_MODEL|NC_NETCDF4, &ncid)) ERR;


/* Define dimensions in order. */
if (nc_def_dim(ncid, DIM0, NC_UNLIMITED, &dimids[0])) ERR;
if (nc_def_dim(ncid, DIM1, 4, &dimids[1])) ERR;


/* Define coordinate variables in a different order. */
if (nc_def_var(ncid, DIM1, NC_DOUBLE, 1, &dimids[1], &varid[1])) ERR;
if (nc_def_var(ncid, DIM0, NC_DOUBLE, 1, &dimids[0], &varid[0])) ERR;
```

In the above case the order of the coordinate variables is different from the order of the dimensions.

This practice should make little difference in user code. If the user is writing code that depends

14

on the ordering of dimensions, the netCDF library was updated in version 4.1 to detect this condition, and add the attribute _Netcdf4Dimid to the dimension scales in the HDF5 file. This attribute holds a scalar H5T_NATIVE_INT which is the (zero-based) dimension ID for this dimension.

If this attribute is present on any dimension scale, it must be present on all dimension scales in the file.

B-6   Variables

Variables in netCDF-4/HDF5 files correspond exactly to HDF5 datasets. The data types match naturally between netCDF and HDF5.

In netCDF classic format, the problem of endianness is solved by writing all data in big-endian order. The HDF5 library allows data to be written as either big or little endian, and automatically reorders the data when it is read, if necessary.

By default, netCDF uses the native types on the machine which writes the data. Users may change the endianness of a variable (before any data are written). In that case the specified endian type will be used in HDF5 (for example, a H5T_STD_I16LE will be used for NC_SHORT, if little-endian has been specified for that variable.)

NC_BYTE

    H5T_NATIVE_SCHAR

NC_UBYTE

    H5T_NATIVE_SCHAR

NC_CHAR

    H5T_C_S1

NC_STRING

    variable length array of H5T_C_S1

NC_SHORT

    H5T_NATIVE_SHORT

NC_USHORT

    H5T_NATIVE_USHORT

NC_INT

    H5T_NATIVE_INT

NC_UINT

    H5T_NATIVE_UINT

ESDS-RFC-022v1   **Edward Hartnett**
Category: Standards Track   **March 2011**
Updates/Obsoletes: None   **netCDF-4/HDF5 File Format**

NC_INT64

H5T_NATIVE_LLONG

NC_UINT64

H5T_NATIVE_ULLONG

NC_FLOAT

H5T_NATIVE_FLOAT

NC_DOUBLE

H5T_NATIVE_DOUBLE

The NC_CHAR type represents a single character, and the NC_STRING an array of characters. This can be confusing because a one-dimensional array of NC_CHAR is used to represent a string (i.e. a scalar NC_STRING).

An odd case may arise in which the user defines a variable with the same name as a dimension, but which is not intended to be the coordinate variable for that dimension. In this case the string "_nc4_non_coord_" is pre-pended to the name of the HDF5 dataset, and stripped from the name for the netCDF API.

B-7   Attributes

Attributes in HDF5 and NetCDF-4 correspond very closely. Each attribute in an HDF5 file is represented as an attribute in the netCDF-4 file, with the exception of the attributes below, which are ignored by the netCDF-4 API.

_Netcdf4Coordinates

An integer array containing the dimension IDs of a variable which is a multi-dimensional coordinate variable.

_nc3_strict

When this (scalar, H5T_NATIVE_INT) attribute exists in the root group of the HDF5 file, the netCDF API will enforce the netCDF classic model on the data file.

REFERENCE_LIST

This attribute is created and maintained by the HDF5 dimension scale API.

CLASS

This attribute is created and maintained by the HDF5 dimension scale API.

DIMENSION_LIST

This attribute is created and maintained by the HDF5 dimension scale API.

ESDS-RFC-022v1   **Edward Hartnett**
Category: Standards Track   **March 2011**
Updates/Obsoletes: None   **netCDF-4/HDF5 File Format**

NAME

This attribute is created and maintained by the HDF5 dimension scale API.

B-8   User-Defined Data Types

Each user-defined data type in an HDF5 file exactly corresponds to a user-defined data type in the netCDF-4 file. Only base data types which correspond to netCDF-4 data types may be used. For example, no HDF5 reference data types may be used.

B-9   Compression

The HDF5 library provides data compression using the zlib library and the szlib library. NetCDF-4 only allows users to create data with the zlib library (due to licensing restrictions on the szlib library). Since HDF5 supports the transparent reading of the data with either compression filter, the netCDF-4 library can read data compressed with szlib (if the underlying HDF5 library is built to support szlib), but has no way to write data with szlib compression.

With zlib compression (a.k.a. deflation) the user may set a deflation factor from 0 to 9. The zero deflation level does not compress the data, but does incur the performance penalty of compressing the data. The netCDF API does not allow the user to write a variable with zlib deflation of 0. It turns off deflation for the variable instead. NetCDF can read an HDF5 file with deflation of zero, and correctly report that deflation factor to the user.

B-10   The NetCDF-4 Classic Model Format

Every classic and 64-bit offset file can be represented as a netCDF-4 file, with no loss of information. There are some significant benefits to using the simpler netCDF classic model with the netCDF-4 file format. For example, software that writes or reads classic model data can write or read netCDF-4 classic model format data by recompiling/relinking to a netCDF-4 API library, with no or only trivial changes needed to the program source code. The netCDF-4 classic model format supports this usage by enforcing rules on what functions may be called to store data in the file, to make sure its data can be read by older netCDF applications (when relinked to a netCDF-4 library).

Writing data in this format prevents use of enhanced model features such as groups, added primitive types not available in the classic model, and user-defined types. Features of the netCDF-4 formats that do not require additional features of the enhanced model, such as per-variable compression and chunking, efficient dynamic schema changes, and larger variable size limits, offer potentially significant performance improvements to readers of data, without requiring program changes.

When a file is created via the netCDF API with a CLASSIC_MODEL mode flag, the library creates an attribute (_nc3_strict) in the root group. This attribute is hidden by the netCDF API, but is read when the file is later opened, and used to ensure that no enhanced model features are written to the file.