Re-engineering HDF5 OPeNDAP handler

Status of This Memo

This is the technical note on re-engineering the HDF5 OPeNDAP handler.

Author and Date

MuQun Yang and Hyo-Kyung Lee

May 2012

Abstract

The HDF5 OPeNDAP handler has been in operational use by NASA data centers to serve HDF-EOS5 data for a few years. Significant limitations have been discovered as new versions of HDF-EOS5 data are distributed. It is also extremely difficult to add the CF support for new NASA HDF5 products in the existing handler. Furthermore, the long-term maintenance cost to support these products in the existing handler will be very costly. So providing better support for NASA HDF5 products with the minimum maintenance cost requires the re-engineering of the handler. This technical note documents the design, implementation, testing, limitations and the future plan of the re-engineered handler.

Table of Contents

1. Introduction

1.1 Overview of HDF5 and OPeNDAP

Hierarchical Data Format Version 5 (HDF5) [1, 2] is a general-purpose library and file format for storing, managing, archiving, and exchanging scientific data. The HDF5 data model includes two primary types of objects, a number of supporting object types, and metadata describing how HDF5 files and objects are to be organized and accessed. The HDF5 file format is self-describing in the sense that the structures of HDF5 objects are described within the file.

Data served using Data Access Protocol Version 2 (DAP2) [3, 4] use simple descriptions built of basic computer datatypes. In a typical deployment, DAP2 servers are written to transform local representations of data (i.e., data stored in specific data formats) into the DAP2 data model. Because each data format has been developed in concert with a certain set of requirements, each format requires special attention when DAP2 servers map the local representation into the DAP2. Throughout the rest of this note, the term 'OPeNDAP' represents client-server software that supports DAP2.

1.2 How to access HDF5 data via OPeNDAP

Figure 1 shows how to access HDF5 files remotely via OPeNDAP. The complete data-processing system consists of six components: (1) the remote HDF5 data (in this example, the HDF5 data are HDF-EOS5 files generated from instruments on the NASA Aura satellite), (2) the HDF5 OPeNDAP handler to map the remote data into DAP2, (3) an OPeNDAP data server (e.g., Hyrax), (4) the networking infrastructure to transport the data encoded using DAP2's data, (5) the client-side DAP2 software to decode the data read from the data server, and (6) a visualization and analysis tool (e.g., IDV) built with the OPeNDAP client library to visualize and analyze the remote HDF5 data.

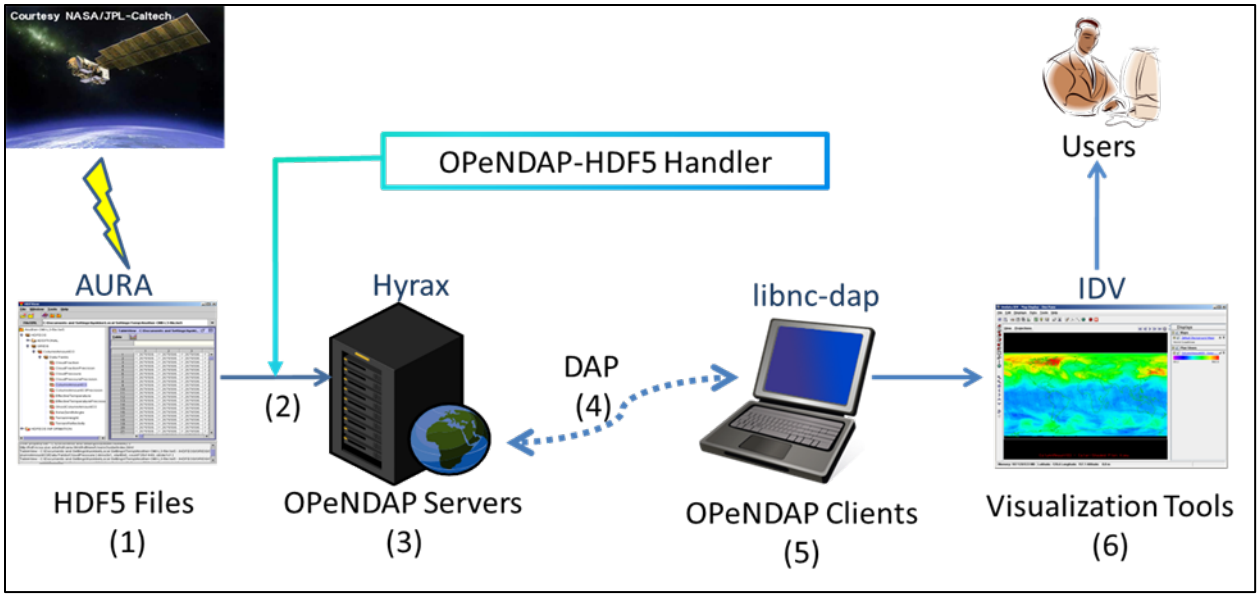

Figure 1. Components of the process for accessing HDF5 files via OPeNDAP

1.2 Motivation

The HDF5 OPeNDAP handler has been in operational use by NASA data centers to serve Aura HDF-EOS5 products. A few Making Earth System Data Records for Use in Research Environments (MEaSUREs) data products also use either HDF5 or HDF-EOS5 as their storage formats. The future Decadal Survey products such as Soil Moisture Active Passive (SMAP) and the Ice, Cloud, and land Elevation Satellite-2 (ICESat-2) will also use HDF5 as their storage format.

However, for multiple HDF-EOS5 swath and grid products, it is almost impossible for the old handler (version 1.5.1 and earlier) to support the CF conventions [5]. Furthermore, the inherent limitation in the old handler makes it extremely difficult to engineer the CF support for the new products. Even if the support can be jammed into the handler, the possible ad-hoc fixes will make the future maintenance extremely challenging. That is why we want to re-engineer the HDF5 OPeNDAP handler so that the new handler not only can support the current and new HDF5/HDF-EOS5 products but also can make the future maintenance much easier.

2. Highlights

The highlights of the re-engineered handler are as follows:

- Upgrade the handler with HDF5 1.8 APIs.
- Re-engineer the CF option to support different NASA products.
    - Support multi-grid and multi-swath HDF-EOS5 products.

    - Support MEaSUREs SeaWiFS, MEaSUREs Ozone, Aquarius, GOSAT/acos, SMAP HDF5 products.
- Re-engineer the CF option to support the long-term maintenance.
    - Modularize the handler to make bug fixes and new product support easier.
    - Implement a comprehensive testsuite.

3. Core engineering

3.1 The limitations of the old handler

The following lists the limitations of the old HDF5 OPeNDAP handler.

- The old handler mixes the CF support with the default behavior. It is extremely difficult to fix bugs or to add new features.
- The old hander follows the CF conventions in an ad-hoc way. It is difficult for the handler to extend its CF support to new HDF5 products.
- The old handler provides limited support for HDF-EOS5 products that have multiple swaths and grids.
- The old handler doesn't follow the CF conventions for non-HDF-EOS5 products. This makes the popular OPeNDAP visualization tools fail to access non-HDF-EOS5 products such as SMAP, ICESat-2, Aquarius, and MEaSUREs.
- The old handler still uses the out-of-date HDF5 1.6 APIs whereas The HDF Group encourages users to use HDF5 1.8 APIs.
- The old handler doesn't map all HDF5 datatypes to DAP2.
- The old handler doesn't support HDF5 cyclic groups and external links.

3.2 Requirements for the CF option

3.2.1 CF conventions

The handler needs to follow the CF conventions for both NASA HDF-EOS5 and generic HDF5 products so that popular OPeNDAP visualization tools (e.g., IDV and Panoply) can visualize these products.

3.2.2 Long-term maintenance

The handler should be modularized to make the bug fixes and the supporting of new products easier.

3.2.3 Object name conventions

We communicated with OPeNDAP service providers at the NASA GES DISC data center to come up with the following object name conventions.

- In general, for any character not allowed by the CF name conventions [7], change that character to underscore ('_').
- In general, variable names of an HDF-EOS5 multi-grid/multi-swath/multi-zonal-average file should have the corresponding grid/swath/zonal-average names prefixed before the field

names. Variable names of an HDF-EOS5 single grid/swath/zonal-average should just use the corresponding field names.

- In general, variable names for any non-HDF-EOS5 files should have their group path prefixed before the HDF5 dataset names. For the supported NASA HDF5 products, if an HDF5 dataset is under the HDF5 root group, the HDF5 dataset name should be used as the variable name. The root group path should not be prefixed.
- The original HDF5 dataset path and names should be preserved by using DAP2 attributes.
- The handler should provide an option to handle object name clashings.

3.3 Design of the re-engineered handler

3.3.1. Overall

3.3.1.1. Separate the CF option from the default option

To ease the long-term maintenance, we totally separate the CF option from the default option in the design.

3.3.1.2. Run-time switch

To make the old handler support the CF option, one needs to configure the handler differently and then compile the source code. The **#ifdef** macro is used inside the source code to support the CF option. This makes the code difficult for people to understand.

The OPeNDAP's BES supports a feature called BES *key* that allows the handler to change the behavior at run-time. These *key*s can essentially provide the same functionality that the **#ifdef** macro can provide. In the meantime, removing the code blocks within **#ifdef** macro can make the code easy to follow. So we decide to use a BES *key* to make the run-time switch between the CF option and the default option. BES *key*s are also used for other run-time switches such as turning on and off the name clash handlings.

3.3.2. Default option

Since the NASA data centers are interested in using the CF option to make data accessible by popular visualization tools, the default option is not our focus. Moreover, after some investigations, we found that our current resources are not sufficient to re-engineer the default option extensively. So we decide to just update the code with HDF5 1.8 APIs. The rest of the implementation for the default option is mostly kept as is.

3.3.3. CF option

This is the core part of the handler. Since OPeNDAP expects to receive metadata and raw data from the handler separately, we divide this section into two parts: Generate DAP2 metadata and Generate DAP2 raw data.

3.3.3.1 Generate DAP2 metadata

OPeNDAP uses Data Descriptor Structure (DDS) and Data Attribute Structure (DAS) to store the metadata. So the handler needs to generate DDS and DAS based on the information transformed from the metadata stored in an HDF5 file.

Figure 2 shows how the handler processes the HDF5 metadata for generating DDS and DAS. The rightmost part of the figure shows the process of handling the metadata for the default option. It follows the default HDF5 to DAP2 mapping [6]. The rest part of the figure shows how the handler follows the CF conventions to generate DDS and DAS. First, we decide to just check if the HDF5 file is an HDF-EOS5 file or a general HDF5 file. However, we also leave a module container in the code (TBD in the figure) to support possible future HDF5 profilers such as JPSS HDF5 products.

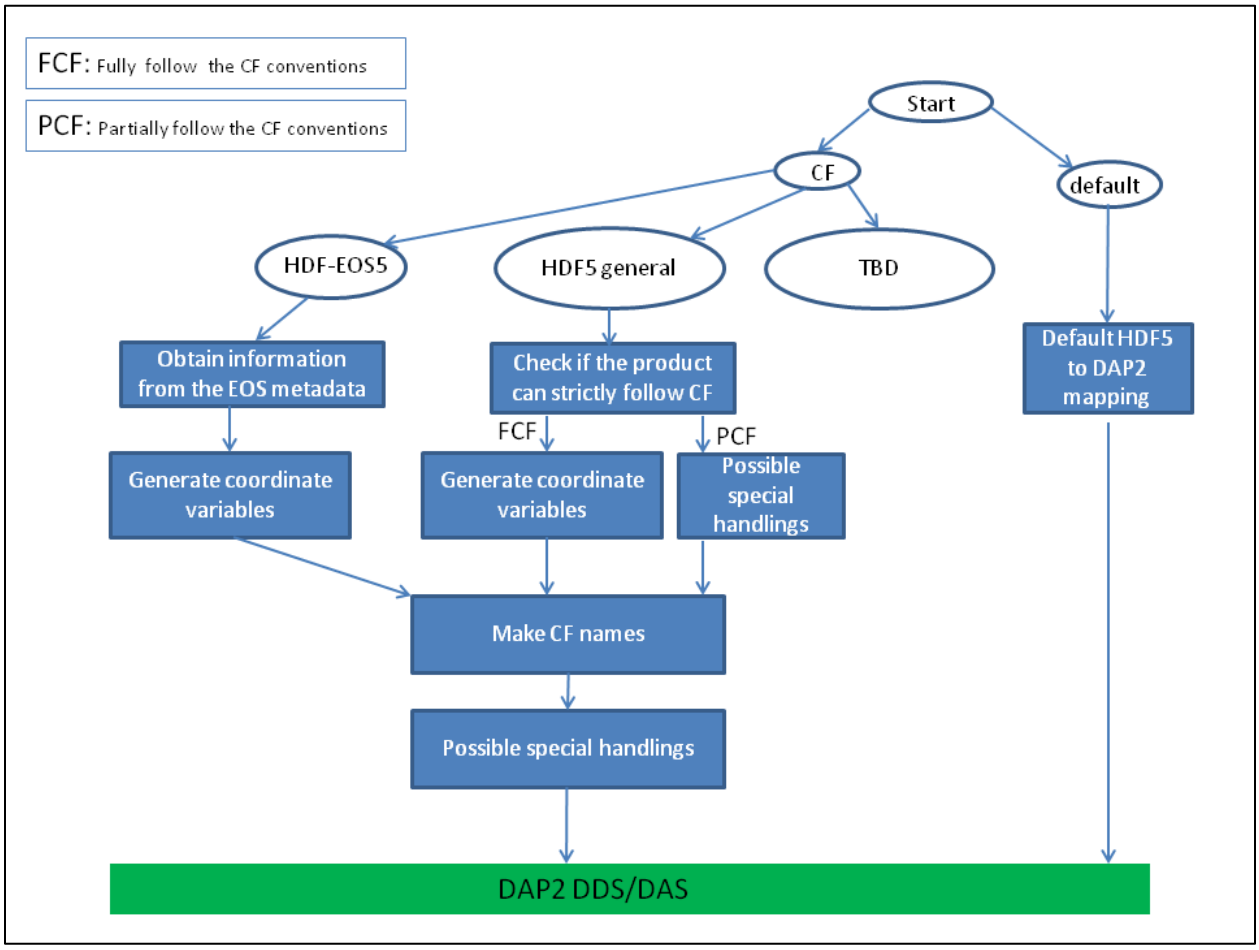

Figure 2. DAP2 DDS and DAS flow chart of the re-engineered HDF5 OPeNDAP handler

For an HDF-EOS5 file, the handler obtains the dimension and other metadata information of the file through a parser. The handler then follows the CF conventions to process the parsed results to prepare for the generation of the coordinate variables.

For a general HDF5 file, we check if this is a NASA HDF5 product that requires the CF support based on the requests from NASA data centers. The current supported products are MEaSUREs SeaWiFS, MEaSUREs Ozone, Aquarius L3m, GOSAT/acos, and SMAP (simulation data). The handler identifies these products by checking HDF5 attributes unique to a particular product. Some NASA products cannot be handled to fully follow the CF conventions. For these products, we discussed with the corresponding

data center on how to handle them. Special handlings are made based on the recommendation from the data center. Often, these products can be made to partially follow the CF conventions. For products that can be translated to fully follow the CF conventions, the handler manipulates the metadata information to prepare for the generation of the coordinate variables. To achieve this goal, different products may need to be handled differently.

Then the handler needs to generate coordinate variables based on the processed metadata. After generating the coordinate variables, object names of all HDF5 files are manipulated to follow CF name conventions [7]. This step is shared by both HDF-EOS5 and general HDF5 files. The object name conventions requested by NASA GES DISC (section 3.2.3) are also fulfilled in this stage.

After making the CF names, the handler may need to add CF attributes such as 'coordinates' for some products to be visualized by OPeNDAP visualization tools. Other special handlings such as adding an attribute to preserve the original HDF5 group path of an object are also done in this stage.

The final stage of this process is to generate DAP2 DDS and DAS.

3.3.3.2 Generate DAP2 Raw Data

The DAP2 raw data generation is totally separated from the metadata generation in order to keep the code modularized. Figure 3 shows the process on how the handler retrieves raw data from HDF5 and passes the data to DAP2. The rightmost part of the figure shows the process of handling the raw data for the default option. It simply follows the standard way to obtain the HDF5 data via HDF5 APIs. The rest part of the figure shows how the handler retrieves the raw data for the CF option.

For the CF option**,** values of some variables cannot be retrieved by simply calling HDF5 APIs. So these variables need to be handled differently. For example, there are no latitude and longitude variables in some HDF-EOS5 grid files. Instead, the latitude and longitude values need to be calculated with the parameters stored in the metadata. The handler retrieves these parameters and follows the corresponding rule to calculate the latitude and longitude values and pass them to OPeNDAP.

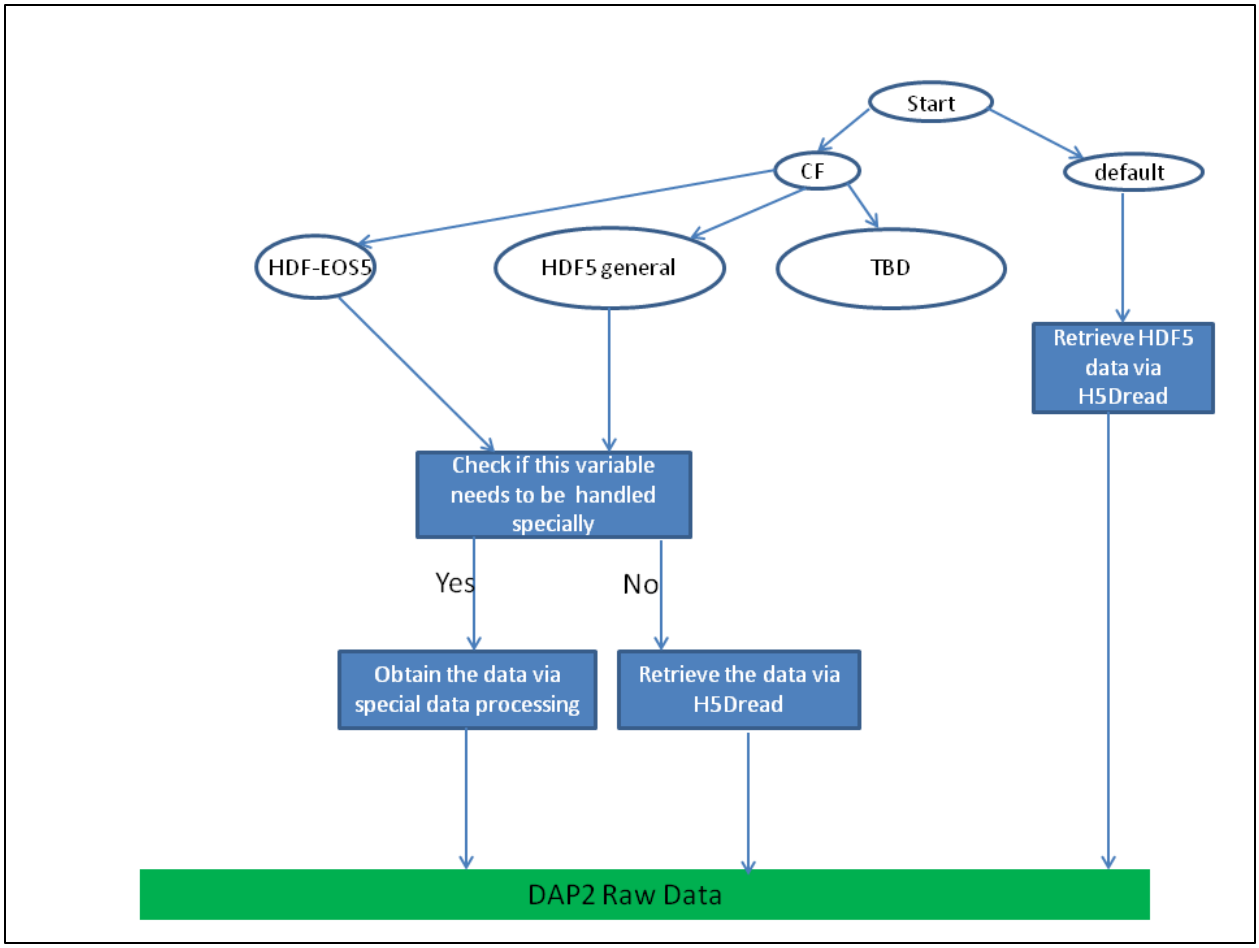

Figure 3. DAP2 raw data flow chart of the re-engineered HDF5 OPeNDAP handler

3.4 Implementation of the re-engineered handler

The implementation largely follows the design. To be consistent with the DAP2 core implementation, we still implement the handler in C++. The C++'s inheritance feature makes the implementation easy to achieve the modularization goal. The C++ Standard Template Library (STL) is also convenient to use for translating the HDF5 metadata to DAP2 DDS and DAS.

Here are a few highlights for the implementation.

- The implementation of the CF option is separated from that of the default option.
- The HDF5 1.8 APIs are used to retrieve HDF5 object information for both the CF and the default options.
- The CF option only:
    - HDF5 products are categorized and are separately handled except for the modules that can be shared. One such example is the module that makes the object names follow the CF name conventions [7].
    - Translating metadata to DAP2 is separated from retrieving the raw data.

- The handler provides an option to handle object name clashings.
- BES *key*s are used to replace the **#ifdef** macro. This makes the code much cleaner and easier to maintain.
- The DAP2 variable and attribute names strictly follow the object name conventions listed in section 3.2.3.

3.5 Documentation of the re-engineered handler

We use Doxygen to provide some useful information of the C/C++ functions and the C++ classes in the header files.

3.6 Limitations of the re-engineered handler

3.6.1 CF option

3.6.1.1 HDF5 objects

3.6.1.1.1 Datatypes

The mappings of 64-bit integer, time, enum, bitfield, opaque, compound, array, and reference types are not supported. Except one dimensional variable length string array, the mapping of the variable length datatype is not supported either. The handler simply ignores these unsupported datatypes.

3.6.1.1.2 Cyclic groups

HDF5 files containing cyclic groups are not supported. If such files are encountered, the handler hangs with infinite loops.

3.6.1.1.3 Links and Comments

The handler ignores soft links, external links and comments. A hardlink is handled as an HDF5 object.

3.6.1.1.4 Dataspace

For the HDF5 datasets created with the scalar dataspace, the handler can only support the string datatypes. It ignores the datasets created with other datatypes. HDF5 allows the size of a dimension to be 0 (zero) for a dataspace. The handler also ignores the datasets created with such dataspace. The mapping of any HDF5 datasets with NULL dataspace is also ignored.

3.6.1.2 NASA Products

Currently, GOSAT/acos and OMI level 2G products cannot be visualized by OPeNDAP visualization tools because of the limitations of the current CF conventions and visualization tools.

We found object reference attributes in several NASA products. Since these attributes are only used to generate the DAP2 dimensions and coordinate variables, ignoring the mapping of these attributes doesn't lose any essential information for OPeNDAP users.

GOSAT/acos product has 64-bit integer objects that DAP2 doesn't support. We follow the suggestion from NASA GES DISC to divide the 64-bit integer into two 32-bit integers and map them to DAP2.

We haven't found other unsupported cases listed in section 3.6.1.1 among NASA products we support.

3.6.1.3 Source code

The current source code doesn't use the OPeNDAP's **BESDebug** function for easy debugging. Also there is no Doxygen support for the source files.

3.6.2 Default option

The mappings of HDF5 64-bit integer, time, enum, bitfield, and opaque datatypes are not supported. Except one dimensional HDF5 variable length string array, HDF5 variable length datatype is not supported either. HDF5 external links are ignored. The mapping of HDF5 objects with NULL dataspace is not supported.

4. Testing

4.1 The testsuite design

We design the testsuite of the handler with an important principle: modularization. We carefully consider possible scenarios and group them into modules.

Figure 4 shows a diagram on how the testsuite is organized. At the top level, we separate the CF option from the default option.

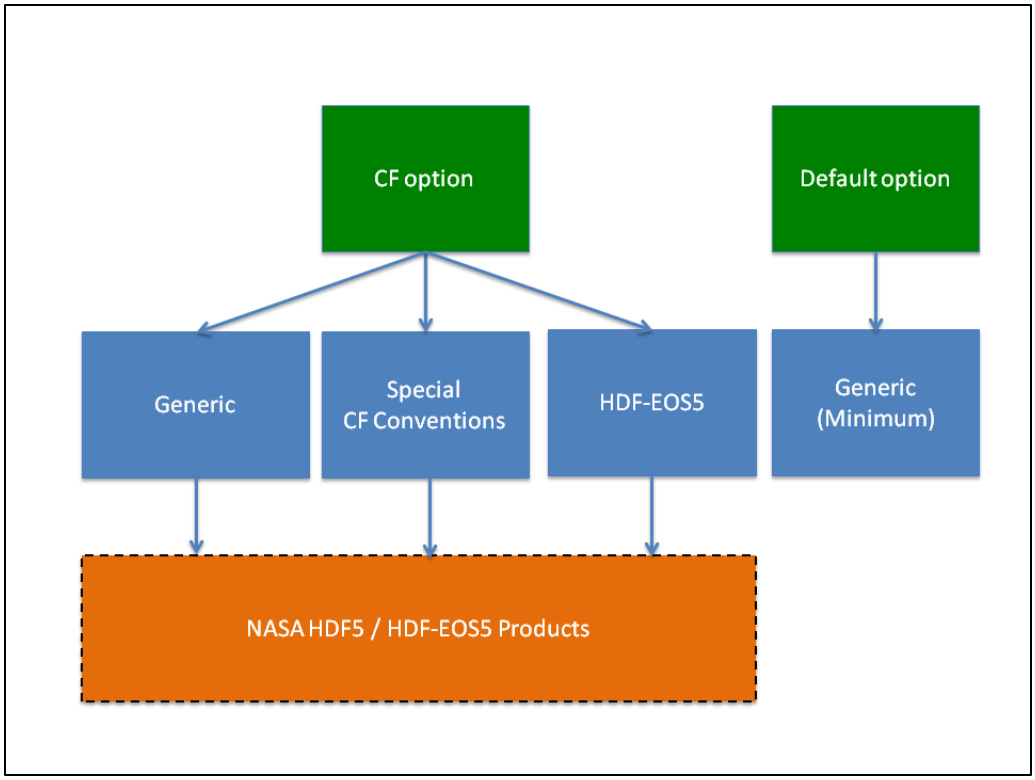

Figure 4. Testsuite Organization Diagram

As shown on the left side of the diagram, we divide the CF option testsuite into three parts: the generic HDF5 module, the special CF feature module, and the HDF-EOS5 module. The generic HDF5 module includes tests to check if all supported HDF5 datatypes are mapped correctly to DAP2. We also test if the handling of unsupported objects corresponds with the limitations listed in section 3.6. The special CF feature module checks if the HDF5 object names follow the object name conventions listed in section 3.2.3. It also checks if the handling of the name clashing works correctly. The HDF-EOS5 module checks if the metadata information for supported HDF-EOS5 products are correctly retrieved. Optionally, the NASA HDF5/HDF-EOS5 product module can be used to check if NASA files are handled according to the requirements listed in section 3.2.

The rightmost of the figure shows the simplified structure of the testsuite for the default option. It only includes the limited tests not covered by the generic HDF5 module in the CF option testsuite. For example, the mapping of HDF5 compound datatype to DAP2 and the handling of HDF5 hard links and soft links are checked in this part of the testsuite.

4.2 The testsuite implementation

Our implementation follows the design mostly. Here are a few highlights for the implementation:

- We provide a full set of source codes for all testing files.

- We have several shell scripts that can be used to selectively test different modules of the handler. For example, one can choose to test either the CF option or the default option. Currently, these scripts are not included in the official release.
- NASA files are grouped into priority levels based on our knowledge about the importance of the products. Currently, these files are not distributed in the official release.

5. Future plan

5.1 Core engineering

5.1.1 CF option

Depending on the funding availability and the request, we plan to support the following:

- Other HDF5 products
- The HDF5 scalar dataspace dataset
- The handling of HDF5 NULL dataspace attributes
- The handling of zero-size dataspace attributes
- HDF5 dimension scale model
- HDF5 links
- HDF5 compound datatype
- The handling of cyclic HDF5 groups

Since both CF conventions and visualization tools evolve, the handler may need to evolve in other aspects for the CF option in the future.

5.1.2 The default option

Code structure may need to be re-organized depending on the funding availability and the request.

5.2 Testing

Depending on the funding availability and the request, we plan to support the following:

- Add tests for new NASA HDF5/HDF-EOS5 products.
- Add the automatic testing with the OPeNDAP clients. Currently, we have to manually test if OPeNDAP clients can access NASA HDF5/HDF-EOS5 products. Making this manual testing process automatic can significantly reduce the maintenance cost.

5.3 Documentation

We plan to add comments for the source files in Doxygen style.

References

[1] ESDS-RFC-007 HDF5: http://earthdata.nasa.gov/library/esds-rfc-007v1-pdf
[2] HDF5 website: http://www.hdfgroup.org/HDF5/
[3] ESDS-RFC-004 The Data Access Protocol – DAP 2.0:
http://earthdata.nasa.gov/sites/default/files/esdswg/spg/rfc/ese-rfc-004/ESE-RFC-004v1.1.pdf
[4] OPeNDAP website: http://opendap.org
[5] NetCDF Climate and Forecast (CF) Metadata Conventions:
http://cf-pcmdi.llnl.gov/documents/cf-conventions/1.6/
[6] ESDS-RFC-017 Mapping HDF5 to DAP2:
http://earthdata.nasa.gov/library/esds-rfc-017v1pdf
[7] The CF naming conventions (the first paragraph of section 2.3 of the CF conventions)
http://cf-pcmdi.llnl.gov/documents/cf-conventions/1.6/

Contacts

MuQun Yang, The HDF Group, myang6@hdfgroup.org

Hyo-Kyung Lee, The HDF Group, hyoklee@hdfgroup.org

Acknowledgement

The authors would like to thank Mr. Daniel Marinelli at NASA EOSDIS for his encouragement and support for the OPeNDAP work at The HDF Group. The authors also thank Dr. Fan Fang, Dr. James Johnson, Dr. Christopher Lynnes, other team members at NASA GES DISC and Ms. Rosanna Sumagaysay at JPL for their valuable suggestions regarding the CF support for NASA HDF5 products. Mr. Barry Weiss provided SMAP simulation HDF5 files for us to test. Mr. James Gallagher and Mr. Patrick West provided technical assistance on how to implement BES *key*s. The authors also appreciated their help.

This work was supported by Subcontract number 114820 under Raytheon Contract number NNG10HP02C, funded by the National Aeronautics and Space Administration (NASA). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Raytheon or the National Aeronautics and Space Administration.