Serving JPSS Data via OPeNDAP

Overview of potential problems and possible solutions

Kent Yang (myang6@hdfgroup.org), Joe Lee (hyoklee@hdfgroup.org)

Prepared for the JPSS Team

12/23/2011

JPSS Project

Version 0.0.1



[www.hdfgroup.org](http://www.hdfgroup.org)

Contents

1. Introduction 3

1.1 Motivation and Purpose 3

1.2 Overview of JPSS Products 3

1.3 Overview of OPeNDAP 3

2. Challenges or Issues 4

2.1 Diversity of JPSS Data Products 4

2.2 Usage of XML Files and References 4

2.3 The HDF5 to DAP2 Mapping Problem 4

2.4 Visualization Problems 4

3. Background 5

3.1 Unsupported Objects 5

3.2 CF-Conventions 5

4. Solutions 6

4.1 CF-Compliancy is not Required 6

4.1.1 Do not Handle External Metadata and External Geo-location Information 6

4.1.2 Handle External Metadata and External Geo-location Information 6

4.2 CF-Compliancy is Required 6

4.2.1 Use JPSS Tools and NetCDF handler 6

4.2.2 Customize HDF5 Handler 6

5. Questions 8

6. Working Plan and Cost Estimation 9

# Introduction

## Motivation and Purpose

This document describes the potential problems of serving JPSS data products via OPeNDAP and provides possible solutions to serve them. Our goal is to provide a rough idea of what would be involved in making OPeNDAP work with JPSS data products.

## Overview of JPSS Products

We have collected some sample JPSS products from the JPSS team and analyzed them under the current OPeNDAP framework. The JPSS data products are quite different from NASA HDF-EOS5 data products in several ways.

* Some key metadata/geo-location information is not stored in the HDF5 file. Some key metadata information is stored in the external product XML file. Geolocation information is stored in another HDF5 file in a non-packaged product.
* JPSS data products use HDF5 object and region references extensively. The HDF5 object and region references are used to describe JPSS products and locate raw data; handling the references by OPeNDAP is critical for JPSS files.
* JPSS data products use data types such as 64-bit (un)signed integers.

## Overview of OPeNDAP

OPeNDAP is a framework that makes data easily accessible to remote locations through OPeNDAP clients such as IDV, Panoply, and GrADS. One key benefit of using OPeNDAP clients is the ability of selecting data from a dataset in a remote file and visualizing the selected data on a map easily.

The OPeNDAP framework is built upon the Data Access Protocol (DAP) 2.0 [1] which defines a specific set of data types that can be served by OPeNDAP servers. Therefore, it is not possible to serve HDF5 data directly via OPeNDAP without transforming the original HDF5 objects to objects that the DAP can understand. The *HDF5 handler* module is an OPeNDAP server module that can read objects from HDF5 files and transform them into DAP objects.

# Challenges or Issues

The features of the JPSS data products discussed in section 1.2 makes it hard to serve them with the current HDF5 handler. We list the problems based on the common purpose of serving scientific data via OPeNDAP. If the JPSS data service provider does not care about the completeness and the usability of the data being served, the following problems may be unimportant.

## Diversity of JPSS Data Products

There are many JPSS data products. We need to go through examples of all JPSS products to come up with the complete list of new features for the OPeNDAP HDF5 handler because the list in section 1.2 may not be complete.

## Usage of XML Files and References

The usage of extra XML files and object/region references make it difficult to gather the corresponding metadata and geo-location information for the JPSS data. Assembling all missing information from the dependent external files is prerequisite before any HDF5-to-DAP2 mapping is done if the CF-convention compliancy of JPSS data is desired.

## The HDF5 to DAP2 Mapping Problem

The fundamental problem of serving JPSS data via OPeNDAP is that not all HDF5 objects in JPSS files can be mapped to DAP2 objects by the handler. This is a well known problem that applies not only to JPSS data products but also to any HDF5 data products. One can see the complete list of unsupported objects in Table 4 of [2]. For example, JPSS data products use 64-bit integer data but the Data Access Protocol 2.0 does not support such a data type.

## Visualization Problems

Quite a few Earth Science tools have the capability of visualizing and analyzing remote data via OPeNDAP. However, popular visualization tools such as IDV [3] and Panoply [4] can only visualize and analyze the data if the data from the OPeNDAP server follows CF conventions. Otherwise, the final visualization result is often wrong or misleading. For example, the final results may have the wrong units or display data at the wrong location. JPSS data products have little or no information that meets the CF-convention fully. Thus, clients will fail to visualize the data. That does not mean that JPSS data is not accessible via OPeNDAP. Data can still be retrieved with text-based clients such as web browsers.

# Background

We will discuss some previous work related to the problems in section 2.

## Unsupported Objects

The current HDF5 handler throws an error if it sees unsupported objects in an HDF5 file. A similar OPeNDAP module, the NetCDF handler, has an option of ignoring unsupported objects instead of throwing errors [5]. We plan to implement a similar **ignore** option in the new HDF5 handler. One drawback of the ignore option is that there is no way for OPeNDAP clients to figure out that the original data had extra information that was not supported.

The OPeNDAP team plans to release a new version (4.0) of the Data Access Protocol (DAP4). We believe that the new protocol will support all objects found in JPSS products but we do not know when it will be implemented. Furthermore, we also do not know how soon the tools that can access the data via OPeNDAP will support DAP4.

## CF-Conventions

Meeting the CF-Conventions for JPSS is already tested with the local access case. The JPSS augmentation tool [6] and h5edit tool [7] can modify a local JPSS VIIRS file so that NetCDF tools such as IDV and Panoply can access and visualize the JPSS data locally. This is possible because the tools insert and edit all the necessary CF-compliant geo-location and metadata information inside the file.

The current HDF5 handler with a CF configuration option can detect HDF-EOS5 files and perform a similar operation on the fly. For example, the original HDF-EOS5 Grid files do not have coordinate variables (e.g., latitude and longitude), but the HDF5 handler generates them when the data request comes from the clients. This solution is very useful because clients can still visualize the data correctly without modifying the original files.

# Solutions

There are two categories of solutions and two possible solutions under each category to serve JPSS data via OPeNDAP based on the previous sections.

## CF-Compliancy is not Required

The first category is to serve JPSS data without worrying about CF-compliancy or OPeNDAP visualization tools.

### Do not Handle External Metadata and External Geo-location Information

The first solution is to serve the JPSS data as is without opening any external files. This solution simply ignores data that uses unsupported data types.

### Handle External Metadata and External Geo-location Information

The second solution is to handle external files and serve them together with the JPSS file as a package. This solution will allow users to view the attributes of each JPSS dataset inline when they open the remote JPSS file with web browsers. This solution will also add geo-location variables if the original JPSS file does not have them by reading the external HDF5 file. The solution in section 4.1.1 is also required.

## CF-Compliancy is Required

The second category is to serve the JPSS file to meet the CF standards so that OPeNDAP visualization tools can display individual datasets in a file.

### Use JPSS Tools and NetCDF handler

The first solution is to modify the JPSS files with the JPSS tools and serve them with the current NetCDF handler. Although we are not certain about how well the current NetCDF handler can serve the augmented and edited NetCDF4 JPSS files, it should be straightforward and seamless in theory. All OPeNDAP visualization clients that support local NetCDF files should be able to display the remote JPSS data via OPeNDAP as if they are local files.

This solution has one drawback: all JPSS data files would need to be pre-processed by the data service provider such as NOAA. The original JPSS HDF5 files will be modified by the JPSS tools.

### Customize HDF5 Handler

The second solution is to customize the current HDF5 handler to support JPSS files directly. This approach transfers the role of the JPSS tools to the HDF5 handler. This solution will require the handler to open both the JPSS file and the external XML file, ignore or convert unsupported objects, insert and edit any missing CF-compliant variables and attributes, and deliver the newly assembled objects in DAP 2.0.

Although similar work has been done with HDF-EOS5 case, this task is not simple and requires a significant amount of programming effort. Part of the effort will be similar to that of the augmentation tool programming effort. If there are many different JPSS products that should be handled differently by the handler, then the maintenance of a specialized JPSS handler could be costly.

This solution can be inefficient and costly in terms of performance since new CF-compliant information would need to be re-created every time a new request comes from the client. To improve the efficiency of customization, one possible solution is to use NcML extensively to add any missing information that the CF-convention requires on top of the HDF5 handler output. Although this solution requires the creation of an NcML file for each JPSS file being served, it eliminates the burden of modifying files directly. If NcML file creation can somehow be automated through a database query, this will be a quite effective solution. One drawback to this is that we are not certain how robust the current implementation of the NcML module in OPeNDAP framework is and whether it can provide all the required information. Particularly, we are not sure if JPSS products with external references and aggregation will match well with the NcML’s aggregation scheme.

# Questions

There can be many other unexpected pros and cons on the solutions that we presented in section 4.2. In terms of certainty and time, the NetCDF approach in 4.2.1 looks most promising. In terms of preservation of the original data, the customization approach in 4.2.2 is the only solution. In terms of storage usage, the custom approach in 4.2.2 will be the clear winner although it will use more CPU.

This analysis raises several questions about JPSS OPeNDAP handler requirements.

* Users
	1. Who are the target users of the JPSS OPeNDAP service?
	2. Do they need any visualization capabilities from the OPeNDAP server output?
* Time
	1. How soon should the JPSS files be served by OPeNDAP server?
	2. Can data service providers wait until DAP 4.0 is out?
* Data
	1. How essential is it to serve all data including 64-bit integer types?
	2. Which JPSS products are going to be served in public and private?
	3. How diverse are they in terms of handling CF-compliant objects?
	4. Are the aggregated files over space and time going to be served? If so, will they contain geo-location/time information inside the file or stored in the external file?
* Performance
	1. Are computational resources cheaper than storage at data centers?
	2. Is it possible that JPSS files are too big to be served via OPeNDAP efficiently?
	3. Can NcML files be generated directly from a database by the data producers when XML files are processed?

# Working Plan and Cost Estimation

Here is our future plan after we receive feedback from the JPSS team on this document. Both options will be proof-of-concept studies, not a final production-ready product.

* Review feedback, gather requirements, and make a project plan (40 hours)
* Option 4.1.1: Handle unsupported data types only. (20 hours)
* Option 4.1.2: Handle external files. (80 hours)
* Option 4.2.1: Pre-process JPSS files with JPSS tools and then serve with the hdf5\_handler. (40 hours)
* Option 4.2.2: Process JPSS files on demand by customizing the hdf5 handler (160 hours)
	+ Analyze how the JPSS data model and HDF5 in-memory editing can fit into hdf5\_handler. (40 hours)
	+ Implement a prototype JPSS hdf5\_handler. (40 hours)
	+ Test and identify potential issues. (40 hours)
	+ Tweak the existing handler to handle unsupported objects and test with NCML. (40 hours)
* Write another report with investigation results and recommendations. (40 hours)

References

[1] The Data Access Protocol — DAP 2.0: http://www.opendap.org/pdf/ESE-RFC-004v1.2.pdf

[2] Mapping HDF5 to DAP2: http://www.esdswg.org/spg/rfc/esds-rfc-017/ESDS-RFC-017.pdf

[3] Integrated Data Viewer (IDV): http://www.unidata.ucar.edu/software/idv/

[4] Panoply netCDF, HDF and GRIB Data Viewer: http://www.giss.nasa.gov/tools/panoply/

[5] BES – Modules – The NetCDF Handler: http://docs.opendap.org/index.php/BES\_-\_Modules\_-\_The\_NetCDF\_Handler

[6] h5augjpss: http://www.hdfgroup.org/projects/npoess/h5augjpss\_index.html

[7] h5edit: http://www.hdfgroup.org/projects/npoess/h5edit\_index.html