EUMETSAT
Ocean and Sea Ice SAF

Description of the osihdf5 format

Øystein Godøy

22. September 2006

Norwegian
Meteorological Institute
met.no

Versions

Version	Date	Comment	Responsible
1.1	21.09.2006	Added basic description of R-package osisaf and autoconf configuration of library.	Øystein Godøy
1.0	06.10.2005	Software versions updates, minor modifications of header, added "How to" section.	Øystein Godøy
0.2	08.05.2002	Minor updates.	Øystein Godøy
0.1	12.03.2000	First draft.	Øystein Godøy

Table of Contents

1 Introduction

Within the Ocean and Sea Ice SAF[1] (OSI SAF) project at the Norwegian Meteorological Institute (METNO) there was a need for a practical format during the development and for operational storage of data. The SAF concept is obliged to deliver data in WMO GRIB format, but this format is not directly suitable for storage of satellite data and especially of data with mixed properties.

As NCSA[2] HDF[3] had been used for various applications and purposes at METNO before it was decided to continue using this format. It is also slowly becoming a standard in the meteorological community (e.g. within the NORDRAD co-operation of the Nordic countries). The applications of HDF used at METNO earlier was conforming to version 4[4], when the SAF project started version 5[5] was available. The first prototype of the *osihdf* format was developed in HDF5-1.2.2, but this was upgraded to HDF5-1.4.2-patch1 in 2002. The current implementation relies on HDF5-1.4.4.

Access to files stored in *osihdf5* is possible by using the libosihdf5 library. This is a static library lying above the core hdf5 library developed by NCSA.

This document describes the overall structure of the *osihdf5* format and the general characteristics of the libosihdf5 library. For detailed information on how the various OSI SAF products are implemented in *osihdf5* format see the product manual.

2 Implementing *libosihdf5*

2.1 General

Implementation of libosihdf5 is easily done by the following steps. However, remember that NCSA HDF5 is required to implement libosihdf5.

1. Get the tarball from http://saf.met.no/
2. Unpack tarball
3. Check README
4. Use `configure` to generate Makefiles
5. `make`
6. `make install`
7. `make clean` (if disk space is an issue)

[1] Satellite Application Facility

[2] National Center for Supercomputing Applications

[3] Hierarchical Data Format

[4] For further information on the old NCSA HDF see [http://hdf.ncsa.uiuc.edu/hdf4.html]

[5] For further information on NCSA HDF5 see [http://hdf.ncsa.uiuc.edu/HDF5/]

This software only generates archive files, no shared objects are generated.

2.2 How to use `configure`

To get help type:

```
./configure --help
```

To implement the library in another location than `/usr/local` type:

```
./configure –prefix=/disk1/libs/libosihdf5
```

To build library with NCSA HDF5 in a non-standard directory type:

```
./configure –with-hdf5=/disk1/libs/libhdf5-1p6p4
```

2.3 How to use `make`

Makefiles are automatically generated by the `configure` script. They support the following commands:

`make`
: Build the software

`make clean`
: Remove all object and archive files in the `src` directory

`make distclean`
: As `make clean` with the addition that all automatically generated Makefiles and configuration files are removed as well

`make install`
: Installs archive and header files in `lib` and `include` directories of the directory specified through prefix

3 *osihdf5* format structure

Basically the *osihdf5* format is a composition of a data header and one or more two dimensional data fields. The format implementation in the OSI SAF is described in the diagram in Illustration 1.

Below the elements of the header and data fields are discussed, element names refers to the library header file given later in this document.

The Header contains the information required to position the product in space and time as well as to identify the product's nature. At present only Polar Stereographic map projection is supported (as this is the map projection used for High Latitude products). A more detailed description of the various elements in the header is given below:

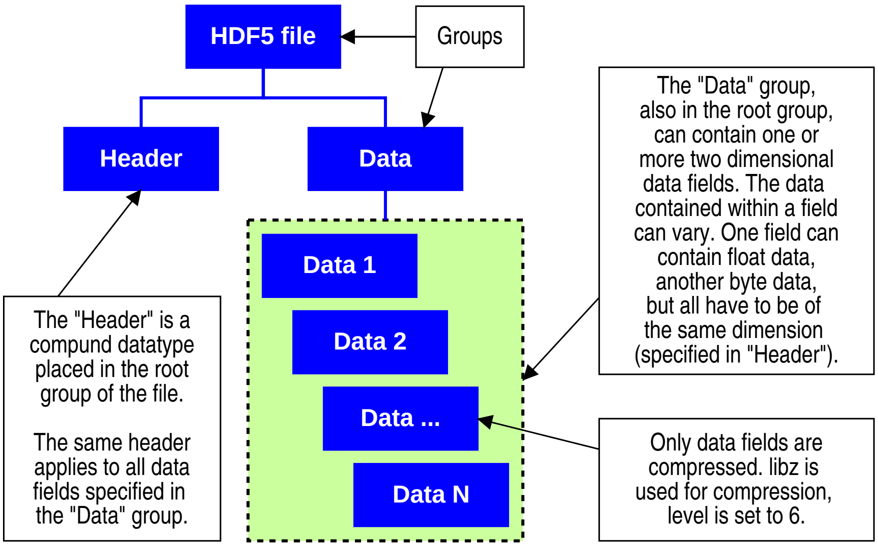
1. char source[32] - string describing the satellite (e.g. NOAA-17) or production software (e.g. OSI SAF)

2. char product[16] - string describing the product (e.g. SST, SSI, DAILY SSI, Ice Conc)
3. char area[16] - string describing the tile used (e.g. nr, ns, at, gr, OSISAF_NH)
4. char projstr[128] - string describing the PROJ.4 map projection specification used
5. unsigned int year - year in 4 digits (e.g. 2002)
6. unsigned short month - month in 2 digits (1-12)
7. unsigned short day - day of month (1-31)
8. unsigned short hour - hour of day (0-23)
9. unsigned short minute -minute of hour (0-59)
10. unsigned int iw - horizontal dimension of products in the data group
11. unsigned int ih - vertical dimension of products in the data group
12. unsigned short z - number of products in the data group
13. float Ax - horizontal resolution in km of a pixel
14. float Ay - vertical resolution in km of a pixel
15. float Bx - distance in km of upper left corner to Greenwich meridian y-axis of Cartesian co-ordinate system[6] (negative westwards)
16. float By - distance in km of upper left corner to North Pole x-axis of cartesian co-ordinate system (negative southwards)

The data fields in the data group is composed of 3 elements:

1. void *data - the 2D data field which can take any data type (float, unsigned char, etc.),
2. char description[16] - a specification of the data type of the 2D field (to help use the data),
3. osi_dtype type - a string description of the 2D data field (describing the product contained, e.g. QFLG or SSI).

[6] The Cartesian co-ordinate system used has origo at the North Pole. The positive x-axis extends eastwards and the negative y-axis extends southwards along the Greenwich meridian.



HDF5 file
Groups
Header
Data
Data 1
Data 2
Data ...
Data N
The "Data" group, also in the root group, can contain one or more two dimensional data fields. The data contained within a field can vary. One field can contain float data, another byte data, but all have to be of the same dimension (specified in "Header").
The "Header" is a compund datatype placed in the root group of the file.
The same header applies to all data fields specified in the "Data" group.
Only data fields are compressed. libz is used for compression, level is set to 6.

Illustration 1 Schematic description of the osihdf5 implementation.

4 The libosihdf5 library

4.1 Introduction

As of 22.05.2006 the operational implementation of osihdf5 within the OSISAF is based upon HDF5-1.6.4, the previous implementation from 05.10.2005 was based upon HDF5 v1.4.4. Prior to that HDF5 1.4.1-patch1 was used. All implementations have been using data chunking and compression of the two dimensional fields. The compression level is hard-coded to 6 and the data chunk size is hard-coded to 100x100 in early realeases, this was changed to 50x50 in the most recent release. There is at present no opportunity to read data as chunked, the whole field will be returned when asked for.

The library libosihdf5 is a static archive defining the intermediate storage format for the Ocean and Sea Ice SAF products. Basis for the format is the NCSA HDF5[7] format description and underlying library.

4.2 Implementation

The data structure reflecting the format is defined as *osihdf5* (see below), containing objects of type *Header* (*PRODhead*), a group of data fields (*datafield*) including a *datatype* specification, and a quality flag[8] (*QFLG*). A specific remark is that the data is contained in a `void *` variable and thus should be type-casted every time it is used. The underlying data type is defined by the `osi_dtype` variable.

[7] http://hdf.ncsa.uiuc.edu/HDF5/

[8] The quality flag fields are yet not implemented in the sea ice products.

4.2.1 Header file

```
typedef enum osi_dtype {
    OSI_UCHAR=0,
    OSI_USHORT,
    OSI_UINT,
    OSI_FLOAT
} osi_dtype;

typedef struct {
    char source[32];
    char product[16];
    char area[16];
    char projstr[128];
    unsigned int year;
    unsigned short month;
    unsigned short day;
    unsigned short hour;
    unsigned short minute;
    unsigned int iw;
    unsigned int ih;
    unsigned short z;
    float Ax;
    float Ay;
    float Bx;
    float By;
} PRODhead;

typedef struct {
    void *data;
    char description[16];
    osi_dtype type;
} datafield;

typedef struct {
    PRODhead h;
    datafield *d;
} osihdf;

/*
 * Function prototypes.
 */
short store_hdf5_product(char *filename, osihdf f);
short read_hdf5_product(char *filename, osihdf *f, short mode);
short init_osihdf(osihdf *f);
short malloc_osihdf(osihdf *f, osi_dtype *dfs, char **desc);
short free_osihdf(osihdf *f);
```

In order to use the library the file [safhdf.h] should be included. For reading it is sufficient to use the function [read_hdf5_product]. This function will allocate the memory required to hold data. Memory can be freed using [free_osihdf]. For writing/creating data structures one should first use [init_osihdf], then [malloc_osihdf] and finally write data using [store_hdf5_product].

4.3 Compression

Compression is implemented using *libz* algorithms on the two-dimensional data fields stored in the file. The header object is not compressed.

A quick inspection reveals that the files that used to take 34 Mb claims from 144 Kb to 9.8 Mb after compression depending on the coverage within the area represented.

4.4 Future

This software has been tested with NCSA HDF5 version 1.6.4.

Furthermore:

- Modifications of the libosihdf5 implementation to make the files readily viewable with e.g. h5view are planned.
- Modifications of the libosihdf5 implementation to allow for a generic interface for map projections are planned.
- Modifications of the libosihdf5 implementation to simplify the handling of various data types through `void *` interfaces are planned.

5 How to

5.1 Read data into C

Use the libosihdf5 software. This software is a C-library available at http://saf.met.no/. It is a high level interface to the NCSA HDF5 library and is distributed with a GNU autoconf setup. See this manual for details.

5.2 Read data into Fortran

5.2.1 General

Use the libosihdf5 software.

5.2.2 Known problems

Interfacing C and Fortran is not directly portable. Generally the largest problems are encountered while transferring character strings (especially several thereof). This should not be necessary for *osihdf5* files.

More TBW.

5.3 Read data into Matlab

5.3.1 General

The following code has been successfully used by Dr. Ole Anders Nøst (Norwegian Polar Institute) to read data from the *osihdf5* files into Matlab for further processing. As can be seen from the code, some processing of the lines is required to get the image correct. This problem is only known from Matlab so far. Possible causes will be examined.

5.3.2 Code sample

```
function dset=readhdf(filename)

dset=hdf5read(filename,'/Data/data[00]');
data=[];
for i=1:630
data=[data,dset(i,1:end)];
end
for i=1:450
data2(i,1:630)=data((i-1)*630+1:i*630);
end
dset=data2;
```

5.4 Read data into R

5.4.1 General

A R-package have been developed. This have some basic functionality for reading and analysing the HDF5 files generated at the OSISAF High Latitude centre. You need the HDF5 and the OSIHDF5 libraries to implement the R-package.

5.4.2 How to install

The R-package will be implemented into the directory pointed at by the environment variable R_LIBS. Remember that if you have not specified R_LIBS yourself, you most probably will need root permissions to implement the package.

Use the following command to install the osisaf package:

```
R CMD INSTALL
```

To get an overview of the interface of this command type:

```
R CMD INSTALL –help
```

As the osihdf5 library most probably is implemented in a non-standard directory use the following command to provide the autoconf setup of the R-package with the proper information:

```
R CMD INSTALL
--configure-args=”--with-hdf5=/disk1/hdf5lib-1p6p4
--with-osihdf5=/disk1/osihdf5lib”
```

5.4.3 How to use

See package within R help for details. The most useful function is `readosisaf(filename,nomiss=TRUE)`. The specified product is returned as a list containing metadata and data.

6 Summary

The most updated information of this format will always be available at the project web page [http://saf.met.no/].