H4toH5 Conversion Library
User's Guide

1. Introduction

Two tools for converting HDF4 files and objects to HDF5 are available, the `h4toh5` utility and the H4toH5 Conversion Library:

- The `h4toh5` utility converts an entire HDF4 file to an equivalent HDF5 file containing equivalent objects with identical content. The conversion is performed according to the default mappings defined in the specification document *Mapping HDF4 Objects to HDF5 Objects*. (Full references for the h4toh5 utility and *Mapping HDF4 Objects to HDF5 Objects* appear in section 7, "References.")
- The H4toH5 Conversion Library, generally referred to as the H4toH5 library, is a set of C library routines that perform part or all of the HDF4-to-HDF5 conversion when called from a C program. The H4toH5 library uses both the HDF4 and HDF5 libraries.

Six basic types of HDF4 objects can be converted to corresponding HDF5 objects:

- SDSs
- Images, including general raster images (GR) and 8- and 24-bit raster images (RI8 and RI24)
- Vdatas
- Vgroups
- Palettes
- Annotations

In each case, the data, attributes, and auxiliary objects, such as HDF4 dimension scales, may be converted. Attributes of the HDF4 objects listed above may be converted to attributes of the corresponding HDF5 objects on a per object basis or individually.

Table 1. Summary of default conversions of HDF4 objects.

HDF4 Object	HDF5 Object	Notes	Ref*
SDS	Dataset	HDF4 SDSs and associated attributes and dimension scales are converted to HDF5 datasets with attributes.	3.1
GR, RI8, and RI24 image	Dataset	HDF4 images and associated attributes and palettes are converted to HDF5-standard images, attributes, and palettes.	3.4
Vdata	Dataset	HDF4 Vgroups, Vdatas, and non-Vgroup members of Vgroups, all with associated attributes, are converted to HDF5 groups and datasets.	3.2
Vgroup	Group		3.3
Annotation	Attribute	HDF4 annotations are converted to HDF5 attributes.	3.6
Palette	Dataset	HDF4 palettes are converted to HDF5 datasets.	3.5
(* Entries in the **Ref** column are section references in *Mapping HDF4 Objects to HDF5 Objects.*)			

2. The H4toH5 Utility versus the H4toH5 Library

The `h4toh5` utility performs a default conversion of the objects in an HDF4 file based on *Mapping HDF4 Objects to HDF5 Objects*. This will be sufficient for many purposes, but many applications may need to convert only selected objects in an HDF4 file. Other applications may need to specify non-default object names or locations in the HDF5 file, creating an HDF5 organizational structure that differs from the structure of the original HDF4 file.

The H4toH5 library provides functions to control such conversion, object by object. This allows an application to convert only selected objects and provides limited controls over how each object is converted. The version 4 of *Mapping HDF4 Objects to HDF5 Objects* specification shows that the H4toH5 SDS dimension conversion should follow the HDF5 dimension scale specification. The `h4toh5` utility was updated to follow this new requirement. However, the H4toH5 SDS conversion APIs still follow the version 3 of the *Mapping HDF4 Objects to HDF5 Objects* to handle the conversion of SDS dimensions. For more information on how HDF4 SDS dimension is mapped in the version 3, please check the Appendix A of the *Mapping HDF4 Objects to HDF5 Objects*.

3. Programming Model

The H4toH5 library is a C library which calls both the HDF4 and HDF5 libraries. The H4toH5 library has APIs and a programming model similar to those of the HDF4 and HDF5 libraries.

There are five steps in the conversion:

1. Initialize the H4toH5 library.
2. Open the HDF4 file and the HDF4 object(s) to be converted.
3. Make appropriate H4toH5 library API calls to perform the conversion.
4. Terminate access to the HDF4 file and the HDF4 object(s), as needed.
5. Terminate the H4toH5 library. This step will close the HDF4 and the HDF5 files.

The following example illustrates the use of H4toH5 library functions to convert an HDF4 SDS to an HDF5 dataset. (The full example appears in the Appendix.)

In this example, one SDS is read from the HDF4 file and a corresponding dataset is created in the HDF5 file. All the dimension scales used by this SDS are converted to HDF5 datasets and all of its attributes are converted to attributes of the corresponding HDF5 dataset.

Step 1. Initialize H4toH5 library interface

```
h4toh5id = H4toH5open(filename4,filename5,H425_CLOBBER);
```

Either the `H4toH5open` or `H4toH5openid` function must be called before any other H4toH5 library function. These calls initialize the H4toH5 library and open the HDF5 file; the application must open the HDF4 file separately.

The arguments specify the source HDF4 file, `filename4`, and the destination HDF5 file, `filename5`. The HDF5 filename can be NULL, in which case, the H4toH5 library will generate a default HDF5 filename. The flag `H425_CLOBBER` indicates that any pre-existing HDF5 file named *`filename5`* is overwritten, i.e., clobbered.

Note that the `H4toH5open` call establishes a context for converting objects from one specific HDF4 file to one specific HDF5 file. To retain this context, *`h4toh5id`* is used in all subsequent calls to the H4toH5 library.

Step 2. Open the HDF4 file and the HDF4 object interface to obtain an identifier for the object to be converted.

As noted above, the application does not need to explicitly open or close the HDF5 file; the H4toH5 library handles that activity. On the other hand, the application does have to explicitly open the HDF4 source file and the HDF4 objects to be converted.

```
sd_id = SDstart(filename4,DFACC_READ);
sds_id = SDselect(sd_id,0);
```

The calling program must make appropriate calls to the HDF4 library to open the HDF4 file and the object to be converted. In this example, the SD interface is used to access the first SDS.

Step 3. Convert the HDF4 object to a corresponding HDF5 object.

```
status = H4toH5sds(h4toh5id, sds_id, "/group1", NULL, NULL,1,1);
```

The `H4toH5sds` function is used here to convert the SDS, referenced by *`sds_id`*, to an HDF5 dataset in the target HDF5 file. The HDF5 dataset will be named after the HDF4 SDS object (`sds_lib1` in this example) and will be a member of the group `/group1`. I.e., the HDF5 dataset will have the absolute path `/group1/sds_lib1`. The user does not have to create the HDF5 `/group1`; the H4toH5 library will automatically create the group as long as the user provides the group name.

The last two arguments specify that all dimension scales used by this SDS object and all HDF4 attributes of this SDS are to be converted. If preferred, selected dimension scales and attributes can be individually converted using the specialized functions provided in the H4toH5 library.

Since the fourth argument is NULL, the dimension scale dataset names in the HDF5 file will be based on the HDF4 SDS dimension scale dataset names and the datasets will be members of the default group `/HDF4_DIMGROUP`.

Step 4. Terminate the access to the HDF4 object and HDF4 file.

```
status = SDendaccess(sds_id);
status = SDend(sd_id);
```

Once all accesses to the HDF4 file have been completed, access to the file must be terminated.

Step 5. Terminate the access to the H4toH5 library.

```
status = H4toH5close(h4toh5id);
```

Once all conversions have been completed, the `H4toH5close` function must be called to release the resources used by H4toH5 library. This call will also close the HDF5 file.

4. Compiling

The H4toH5 library requires both the HDF4 and HDF5 libraries. The application will need to include the H4toH5 library header files `h4toh5.h`. `h4toh5.h` automatically includes the HDF4 and HDF5 library header files.

```
#include "h4toh5.h"
```

The code must be linked with all three libraries (H4toH5, HDF4, and HDF5) as well as any other libraries that may be necessary. See the example `Makefile` in the Appendix.

5. Error handling

A simple error handling mechanism is provided. The application can call `H4toH5error_get` to retrieve an error message. When this routine is called, several level error messages are printed out; errors will be printed starting at the current API and end at the inner-most function where the error was first detected. Errors are summarized in five categories:

1. Memory allocation failed: all memory allocation errors
2. HDF4 internal errors: all HDF4 function errors
3. HDF5 internal errors: all HDF5 function errors
4. Object name handling errors: all errors related to transformation from HDF4 names to HDF5 names
5. Other errors: all other errors

6. Limitations

As outlined below, there are several limitations to the H4toH5 library.

Limited correctness checking

Conversion is complicated and a calling program may potentially request dangerous or incorrect operations, such as converting the two different HDF4 objects into the same HDF5 object. The H4toH5 library cannot comprehensively check whether applications misuse the library.

No recovery

The H4toH5 library cannot recover mistaken converted objects or undo previous conversions. Applications must backup original files.

Limited options

The H4toH5 library provides a few options to control the conversion but cannot offer fine-grained control. Limits include the following:

- Datatype conversions always use the default conversions specified in section 5.5 of *Mapping HDF4 Objects to HDF5 Objects*.

Some objects not converted

Some HDF4 objects may not be converted if they are not in the mapping specification.

Only annotations associated with certain HDF4 objects (SDSs, Vdatas and Vgroups, images, and palettes) are converted to attributes of the corresponding HDF5 objects.

Compression

HDF5 currently supports only gzip and SZIP compression in the standard distribution. HDF4 supports gzip, SZIP, and several additional compression algorithms.

The H4toH5 Conversion Library normally attempts to write HDF4 objects compressed using methods other than gzip or SZIP (e.g., JPEG) as gzip-compressed HDF5 objects.

Raster image interlace mode

The 24-bit raster image line interlace mode currently is not converted.

Dimension scales

Based on the version 4 of the *Mapping HDF4 Objects to HDF5 Objects*, the implementation of the HDF4 SDS dimension scale conversion in the `h4toh5` utility was updated to follow the HDF5 dimension scale specification. However, due to the resource limitation, H4toH5 SDS conversion APIs still follow the version 3 of the *Mapping HDF4 Objects to HDF5 Objects* to handle the conversion of SDS dimensions.

7. References

1. HDF4 to HDF5 mapping specification: Mapping HDF4 Objects to HDF5 Objects
2. HDF4 to HDF5 conversion utility description: h4toh5 utility

Appendix A: Example Makefile and Program

Sample HDF4 file

The sample HDF4 file sds_simple_example.hdf can be found under the /lib/examples of the H4H5 conversion toolkit source release.

Makefile template

This sample Makefile shows how to compile and link a program using the H4toH5 library. Note: if the user configures the toolkit with the optional HDF-EOS2 library(--with-hdfeos2), one also needs to provide the path of the HDF-EOS2 libraries. This sample Makefile assumes that the --with-hdfeos2 option is not used and the user is using the gcc compiler on Linux.

```
## modify paths of HDF5,libh4toh5 and HDF4 to your own path.
HDF4_INSTALL = /usr/local/hdf
HDF5_INSTALL = /usr/local/hdf5
H4TOH5_INSTALL = /usr/local/h4toh5

DEFS =
CFLAGS = $(DEFS) -I$(H4TOH5_INSTALL)/include -
I$(HDF4_INSTALL)/include -I$(HDF5_INSTALL)/include
CC=gcc
LD=gcc

LIBS= $(H4TOH5_INSTALL)/lib/libh4toh5.a
$(HDF4_INSTALL)/lib/libmfhdf.a $(HDF4_INSTALL)/lib/libdf.a
$(HDF4_INSTALL)/lib/libjpeg.a $(HDF5_INSTALL)/lib/libhdf5.a

## If the szip library is used to build HDF4 and HDF5, the szip libary path needs to be included.
# If the user's own zlib library is used to build HDF4 and HDF5,the zlib path needs to be included.
#LIBS= $(H4TOH5_INSTALL)/lib/libh4toh5.a
#$(HDF4_INSTALL)/lib/libmfhdf.a $(HDF4_INSTALL)/lib/libdf.a
#$(HDF4_INSTALL)/lib/libjpeg.a $(HDF5_INSTALL)/lib/libhdf5.a
#$(SZIP_INSTALL)/lib/libsz.a
#$(Zlib_INSTALL)/lib/libz.a

#linux(CentOS 7)
SYSLIBS=-lm -lz -ldl

all: h4toh5example
h4toh5example: h4toh5example.o
 $(LD) -o h4toh5example h4toh5example.o $(LIBS) $(SYSLIBS)

h4toh5example.o:  h4toh5example.c
 $(CC) $(CFLAGS) -c h4toh5example.c
```

Example program

This simple program converts one SDS object from an HDF4 file to an equivalent HDF5 dataset in an HDF5 file.

```
#include "h4toh5.h"

/*
   This example is modified according to test code
h4h5apitestsds.c


*/

/*-------------------------------------------------------------
 * Function: main
 *
 * Purpose:
 *-------------------------------------------------------------
 */


int main() {

  hid_t         h4toh5id;
  int32         file_id;
  int32         sd_id;
  int32         sds_id;
```

```
  /* 1. Initialize h4toh5 library. */
  h4toh5id= H4toH5open("sds_simple_example.hdf","sds_simple_example.h5",H425_CLOBBER);

  /*2. open SD and SDS interface to obtain the id of the first SDS**/
  sd_id   = SDstart("sds_simple_example.hdf",DFACC_READ);
  sds_id = SDselect(sd_id,0);

  /*3. convert the HDF4 object to the HDF5 object.*/
  H4toH5sds(h4toh5id,sds_id,"/group1",NULL,NULL,1,1);

  /*4. Close SDS interface. */
  SDendaccess(sds_id);
  SDend(sd_id);

  /*5. close h4toh5 interface. */
  H4toH5close(h4toh5id);
}
```

Comparison of original HDF4 object and converted HDF5 object

In this section, based on the example program in Appendix A, the `hdp` output of the original HDF4 SDS is compared with the `h5dump` output of the corresponding HDF5 dataset. `hdp` and `h5dump` are, respectively, HDF4 and HDF5 dumping utilities provided for the detailed examination of HDF4 and HDF5 files.

The `hdp` output of the original HDF4 SDS object is as follows:

```
File name: sds_simple_example.hdf

Variable Name = sds_lib1
  Index = 0
  Type= 32-bit signed integer
  Ref. = 2
  Rank = 1
  Number of attributes = 1
  Dim0: Name=dim0
   Size = 4
   Scale Type = 32-bit signed integer
   Number of attributes = 0
  Attr0: Name = sds.attr
   Type = 8-bit signed char
   Count= 10
   Value = test attr1
  Data :
               0 1 2 3


Dimension Variable Name = dim0
  Index = 1
  Scale Type= 32-bit signed integer
  Ref. = 4
  Rank = 1
  Number of attributes = 0
  Dim0: Name=dim0
   Size = 4
  Data :
               0 1 2 3
```

The `h5dump` output of the resulting HDF5 datasets is as follows:

```
HDF5 "sds_simple_example.h5" {
GROUP "/" {
   GROUP "HDF4_DIMGROUP" {
      DATASET "dim0" {
         DATATYPE  H5T_STD_I32BE
         DATASPACE  SIMPLE { ( 4 ) / ( 4 ) }
         DATA {
            0, 1, 2, 3
         }
      }
   }
   GROUP "group1" {
      DATASET "sds_lib1" {
         DATATYPE  H5T_STD_I32BE
         DATASPACE  SIMPLE { ( 4 ) / ( 4 ) }
         DATA {
            0, 1, 2, 3
         }
         ATTRIBUTE "DIMENSION_LIST" {
            H5T_REFERENCE { H5T_STD_REF_OBJECT }
            DATASPACE  SIMPLE { ( 1 ) / ( 1 ) }
```

```
            DATA {
            (0): DATASET 5184 /HDF4_DIMGROUP/dim0
            }
         }


         ATTRIBUTE "DIMENSION_NAMELIST" {
            DATATYPE  H5T_STRING {
                  STRSIZE 276;
                  STRPAD H5T_STR_NULLTERM;
                  CSET H5T_CSET_ASCII;
                  CTYPE H5T_C_S1;
               }
            DATASPACE  SIMPLE { ( 1 ) / ( 1 ) }
            DATA {
               "/HDF4_DIMGROUP/dim0"
            }
         }
   ATTRIBUTE "HDF4_OBJECT_NAME" {
            DATATYPE  H5T_STRING {
                  STRSIZE 8;
                  STRPAD H5T_STR_SPACEPAD;
                  CSET H5T_CSET_ASCII;
                  CTYPE H5T_C_S1;
               }
            DATASPACE  SCALAR
            DATA {
               "sds_lib1"
            }
         }
         ATTRIBUTE "HDF4_OBJECT_TYPE" {
            DATATYPE  H5T_STRING {
                  STRSIZE 3;
                  STRPAD H5T_STR_SPACEPAD;
                  CSET H5T_CSET_ASCII;
                  CTYPE H5T_C_S1;
               }
            DATASPACE  SCALAR
            DATA {
               "SDS"
            }
         }
         ATTRIBUTE "HDF4_REF_NUM" {
            DATATYPE  H5T_STD_U16BE
            DATASPACE  SCALAR
            DATA {
               2
            }
         }
         ATTRIBUTE "sds.attr" {
            DATATYPE  H5T_STRING {
                  STRSIZE 10;
                  STRPAD H5T_STR_SPACEPAD;
                  CSET H5T_CSET_ASCII;
                  CTYPE H5T_C_S1;
               }
            DATASPACE  SCALAR
            DATA {
               "test attr1"
            }
         }
      }
   }
}
}
```

Comparison of the above `hdp` (HDF4) and `h5dump` (HDF5) output yields the following observations:

- The SDS object `sds_lib1` has been converted to an HDF5 dataset named `sds_lib1` in the group `/group1`.
- The user-defined attribute of the SDS object `sds_lib1` has been converted to an HDF5 attribute of the HDF5 dataset.
- The SDS dimension scale dataset `dim0` has been converted to an HDF5 dataset named `dim0` in the group `/HDF4_DIMGROUP`.
- Extra HDF5 attributes can be found in the `h5dump` output of `sds_simple_example.h5` as follows:
  - The object reference attribute `DIMENSION_LIST`
  - The attribute containing the dimension scale name list `DIMENSION_NAMELIST`
  - Attributes containing the HDF4 object type, the HDF4 object name, and the HDF4 reference number

---