Bugs and Suggestions

Please notify us of any bugs you have found in our software and any suggestions you have for future releases or products.

Using the report form below, mail user feedback, bugs, or software suggestions via U.S. mail to:

NCSA Software Tools Group
HDF
152 Computing Applications Bldg.
605 East Springfield Avenue
Champaign, IL 61820

Send reports regarding bugs via electronic mail to:

bugs@ncsa.uiuc.edu

Send reports regarding software suggestions or comments via electronic mail to:

softdev@ncsa.uiuc.edu

Name:

Institution:

Address (Electronic):

Address (U.S. Mail):

Telephone: (_____) _____ – _____.

Version of NCSA HDF: **Type machine:**

Version of system software:

Suggestion or description of problem:

```
  DFSDgetslice:   reads part of a dataset.

  DFSDsettype:    specifies data attributes: data type and
                  representation, system type, and array order.

* new utilities, including the following:

  hdfed:    lets you browse in an HDF file and manipulate some of the
            data
```

Figure F.4 Code Changes: Changes Made to HDF in Release 3.0 and 3.1 (Continued)

```
  fptohdf:  converts floating point data to HDF floating point data
            and/or 8-bit raster images

  r24tohdf: converts a raw RGB 24-bit image to an 8-bit RIS8 with a
            palette

  paltohdf: converts a raw palette to hdf format

  hdftopal: converts palette in an hdf file to raw format
```

```
  DF24getimage: retrieves the image and stores it in an array.

  DF24reqil:   specifies an interlace to be used in place of the
               interlace indicated in the file when the next raster
               image is read.
An interface for annotating HDF data objects and files, which includes
the following routines:
```

Figure F.4 Code Changes: Changes Made to HDF in Release 3.0 and 3.1 (Continued)

```
  DFANgetlablen: gets length of label of a tag/ref

  DFANgetlabel:  gets label of tag/ref

  DFANgetdesclen: gets length of description of tag/ref

  DFANgetdesc:   gets description of tag/ref

  DFANputlabel:  puts label of tag/ref

  DFANputdesc:   puts description of tag/ref

  DFANlastref:   returns ref of last annotation read or written

  DFANlablist:   gets list of labels for a particular tag


An interface for input and output of 8-bit palettes, including the
following routines:

  DFPaddpal:    appends a palette to a file.

  DFPgetpal:    reads in the next palette in the file.

  DFPputpal:    writes a palette to a file.

  DFPnpals:     indicates number of palettes in a file.

  DFPwriteref:  sets the reference number of the next palette to be
                written.

  DFPreadref:   gets the reference number of the next palette to be
                retrieved.

  DFPrestart:   specifies that the next call to DFPgetpal reads the
                first palette in the file, rather than the next.

  DFPlastref:   returns the value of the reference number most recently
                read or written.



Scientific data set routines for storing and retrieving subsets (slices)
of scientific data, and for choosing optional storage formats and data
types:

  DFSDstartslice: prepares system to write part of a dataset to a file.

  DFSDputslice:   writes part of a dataset to a file.

  DFSDendslice:   indicates write completion for part of a dataset.
```

```
    cc myprog.c libdf.a -o myprog

If the include file "dfrig.h" is in the directory "incdir", and the
library file "libdf.a" is in "libdir", use
```

Figure F.4 Code Changes: Changes Made to HDF in Release 3.0 and 3.1

```
#*******************************************************************************
#
#                          NCSA HDF version 3.1
#                                July 1, 1990
#
...
#*******************************************************************************

These are changes made in release 3.1

*       fixed bug concerning checking the status of opening a file
        with unbuffered i/o

*       Added function DF24readref and DFGRreadref for random access
        of 24-bit rasters

*       Added function DF24restart

*       Added function DF24setil

*       Speed up the DFSDgetdata, DFSDputdata, DFSDadddata,
        DFSDgetslice and DFSDputslice functions, especially for UNICOS
        machines

*       Added functions DFANaddfid, DFABaddfds, DFANgetfidlen,
        DFANgetfid, DFANgetdslen, DFANgetfds, DFANaddfann,
        DFANgetfannlen, DFANgetfann and DFANlastref.

*       Revised DFANlablist so that it returns all ref numbers for a
        given tag

*       Fixed bug with DFSDgetdata where it does not move to the next
        SDG

*       Added some macros to make passing character arrays from
        fortran to C easier

*       Fixed some more minor bugs

*       Recoded some parts for cosmetic reasons

_______
New features of HDF 3.0 include the following:

Fortran support for Macintosh II, for Language System Fortran and MPW C
3.0.

An interface for basic i/o of 24-bit raster images, which includes the
following routines:

  DF24addimage:appends a 24-bit raster image set to the file.

  DF24getdims: retrieves the dimensions and interlace of the
               image.
```

```
*****

(Note to Macintosh and PC users:  These routines have been compiled and
run successfully on Macs using MPW C Version 2.0.2, and on PCs using
Lattice C  Version 3.0.  We cannot guarantee that they will compile

correctly with other compilers.  We would appreciate any feedback you
can give on experiences you have compiling them on other compilers.

For a non-Unix system, the Makefile may be used as a guide for compiling
the files.  An approximate summary of the procedure is:

  cc -c df.c dfr8.c dfgroup.c dfcomp.c dfimcomp.c dfsd.c dfkit.c
  ar libdf.a df.o dfr8.o dfgroup.o dfcomp.o dfimcomp.o dfsd.o
  ranlib libdf.a

This creates the library file "libdf.a".

To create the utilities "hdfls", "hdfrseq", "r8tohdf", "hdftor8",
"tektohdf", "hdftotek", and "hdfcomp", the procedure is:

     cc hdfls.c libdf.a -o hdfls
     cc hdfrseq.c libdf.a -o hdfrseq
     cc r8tohdf.c libdf.a -o r8tohdf
     cc hdftor8.c libdf.a -o hdftor8
     cc tektohdf.c libdf.a -o tektohdf
     cc hdftotek.c libdf.a -o hdftotek
     cc hdfcomp.c libdf.a -o hdfcomp

To use the program "hdfseq", create "hdfseq" as a symbolic link to the
executable "hdfrseq".  "hdfseq" displays images on the console of a
Sun or Iris workstation.
```

Figure F.3 INSTALL: Instructions for Installing HDF (Continued)

```
———————— Compiling Subsets of HDF ————————

If you wish to use only some of the HDF Sets, it is possible to create
versions of the library which only contain the desired interfaces.  For
instance, a user who works only with images, but not with raw floating
point data may wish to have only the Raster Image Set (RIS) but not the
Scientific Data Set. The following is the list of source files necessary
for each of the Sets included in the current version of HDF.

Basic Low level HDF:   df.c dfkit.c df.h dfi.h
Basic Low level Fortran: df.c dfF.c dfFf.f dfkit.c df.h dfi.h

8-bit Raster Image Set (RIS-8): dfr8.c df.c dfkit.c dfcomp.c dfimcomp.c
    dfgroup.c df.h dfi.h dfrig.h
8-bit Raster Image Set Fortran: dfr8.c dfr8F.c dfr8Ff.f df.c dfkit.c
    dfcomp.c dfimcomp.c dfgroup.c df.h dfi.h dfrig.h

Scientific Data Set (SDS): dfsd.c df.c dfkit.c dfgroup.c df.h dfi.h dfsd.h
Scientific Data Set Fortran: dfsd.c dfsdF.c dfsdFf.f df.c dfkit.c
    dfgroup.c df.h dfi.h dfsd.h

————————— Compiling C programs with HDF —————————

To use HDF routines in your program, use "#include dfrig.h", "#include dfsd.h"
etc. at the top of your program, depending on the Sets you are using.
Call the appropriate HDF routines as described in the
documentation. Compile your C program "myprog.c" as follows:
```

```
        Fortran stub routines, and utilities
make build — compile library and utilities
make buildnostub — compile library (without Fortran stubs)
        and utilities
make libdf.a — compile library
make libnostub — compile library with Fortran stub routines
make utils — compile utilities
make install — install library and utilities
make clean — rm intermediate files
make cleanup — rm all make products

** VMS
Several DCL script files are provided for compilation.
MAKE.COM — runs MAKELIB.COM and MAKEUTILS.COM
MAKELIB.COM — makes the full library, DF.OLB
MAKENOF.COM — makes the library *without* the fortran
                stubs, also DF.OLB
MAKEUTILS.COM — compiles the utilities

To run MAKE.COM, for example, type @MAKE at the DCL prompt.

Also provided is SETUPUTILS.COM to setup the commands for
the utils to make them easier to use.  Edit this file before
running to customize the directory path.

** MAC
The Macintosh version of HDF is only supported under MPW
3.0, MPW C 3.0, and (if you are using fortran with it) LS
Fortran v2 onwards.

If you have just the generic distribution, the Makefile for
the MAC is in MAKE.HQX, a binhexed version of the MPW
```

Figure F.3 INSTALL: Instructions for Installing HDF (Continued)

```
makefile.  In the Macintosh distribution, the Makefile is
named Makefile.

To generate commands for compiling type
        make
in MPW shell.

Make targets available are :
make / make all — compile and install library and utilities
make allnostub — compile and install library without
        Fortran stub routines, and utilities
make build — compile library and utilites
make buildnostub — compile library (without Fortran stubs)
        and utilities
make libdf.a — compile library
make libnostub — compile library with Fortran stub routines
make utils — compile utilities
make install — install library and utilities
make clean — rm intermediate files
make cleanup — rm all make products

** PC compatibles
The batch file MAKE.BAT is included to compile the library
and utilities.  Type
        MAKE
in the MS-DOS prompt.
```

```
   hdftor8 – a utility to convert an HDF raster 8 bit image into a raw
             raster image.

   r8tohdf – a utility to convert a raw raster image to an HDF raster 8
             bit image.

   hdfcomp – a utility to change the compression scheme used for 8-bit
             raster images in HDF files.

If you execute any of these commands with no parameters, it will display
the list of acceptable parameters.

In the doc/ directory you will find files containing the HDF
documentation.

In the examples/ directory you will find example codes using HDF. The
example codes should be self explanatory.

In the include/ directory you will find the necessary include files for
use in your own programs that link with the HDF library.

In the lib/ directory you will find the HDF library which contains the
necessary subroutines for developing your own HDF applications.

In the src/ directory you will find the source for all of the HDF
utilities and library.

If you have any questions, please contact the NCSA Consulting Office.
```

Figure F.3 INSTALL: Instructions for Installing HDF

```
*************************************************************************

                        NCSA HDF version 3.1
                            July 1, 1990
*************************************************************************

The file HINTS contains some hints for errors and unusual cases.

*Compiling and installing
** UNIX
For UNICOS, SGI or fortran compilers that uses only short
names (< 8 characters), see HINTS.

We now provide makefiles for each configuration we support.
These Makefile's are named Mfile.<SystemName>.  In addition,
Mfile.GEN is the generic makefile which you could modify and
use if your configuration is not supported.

Find the Makefile for your system.  If your system is a sun3
or sun4, you could now do
        cp Mfile.SUN Makefile
        make
or just
        make -f Mfile.SUN
to make the libraries.

Make targets available are :
make / make all — compile and install library and utilities
make allnostub — compile and install library without
```

```
* If you want to see more software like NCSA HDF, you need to send us a
* letter, email or US mail, telling us what you are doing with NCSA HDF.
* We need to know:
* 1) What science you are working on (an abstract of your work would be
* fine);
* 2) How NCSA HDF has helped you, e.g., whether it has increased your
* productivity or allowed you to perform operations you were unable to
* do before.
*
* We encourage you to cite the use of NCSA HDF , and any other NCSA
* software you have used, in your publications.  A bibliography of your
* work would be extremely helpful.
*
* This is a new kind of shareware.  You share your science and successes
* with us, and we attain the resources necessary to share more software
* like NCSA HDF with you.
*
************************************************************************
```

Figure F.2 README.NCSA. for HDF Users of NCSA Computers

```
#*******************************************************************************
#
#                          NCSA HDF version 3.1
#                                July 1, 1990
#
...
#
#*******************************************************************************

*********************************************************************

                       NCSA HDF version 3.1
                           December 1, 1989
*********************************************************************


*****************************
        Using HDF at NCSA
*****************************

 HDF is installed on various systems at NCSA.  The following is a list
of directories in which HDF is installed on different systems:


       Cray-2:      /usr/local/apps/hdf
       Cray-XMP:    /usr/local/apps/hdf
       Suns:        /soft/hdf
       Medusa:      /usr/hdf
       Replicant:   /usr/hdf

The current version of HDF will be in that directory and its
subdirectories.

In the bin/ directory you will find several HDF utilities, such as:

   hdfls - a utility to display information about the contents of an HDF file.

   hdfrseq - a utility to display a raster 8 bit image remotely using
             the NCSA Interactive Color Raster protocol.
```

Figure F.2 README.NCSA. for HDF Users

```
files in src/fixatr also.  If your system is a MacII and you need to use
fortran, get the fortran files in src/mac.  This will prompt you for each
of the source files, asking if you want to download them. Answer "y" to each.
This will produce the source files for that system in your directory.
Compile these files according to the instructions in the file INSTALL.

To obtain the documentation enter "cd ../../doc" to move to the
directory containing the documentation.  There are two subdirectories,
with "ascii" providing the documentation in readable form, and "word"
providing it in Macintosh Microsoft Word format. Each subdirectory has
two subdirectories, containing the user documentation (NCSA_HDF) and the
technical specification (HDF_Specs), respectively.  The Word files must
be downloaded in binary mode with Macbinary mode enabled.  The ascii
files may be downloaded in ascii mode.


*********************************************************************
                Obtaining HDF using remote login
*********************************************************************

If you have an account on the NCSA Suns, you may download HDF in this way.
To obtain a copy of HDF for a particular system, login to zaphod.ncsa.uiuc.edu,
cd to the directory /sdg/ftp/HDF and use the "transfer" script.

Usage: transfer systemtype hostname [directory]

where systemtype is "unicos", "sun", "alliant", "iris4", "mac",
"vms" or "pc", hostname is the ftp name of the host you want to transfer
the files to, and directory is the directory on the target system in
which you want the files to be placed.

Transfer will create the source files appropriate for the system type,
then open an ftp connection to the target machine and ask you to login.
When you do, it will automatically copy the required files to the target
system in the directory you specified.  It will also deposit all the
documentation, in Macintosh Microsoft Word format if transferring to a
Macintosh, in ascii format otherwise.  It will deposit all the files in
the same directory, as contrasted to using the "tar" approach outlined
in the section on anonymous ftp, which will create a tree of
subdirectories.


*_____________________________________
*  * NCSA HDF Version 3.00 source code and documentation are in the
* public domain.  Specifically, we give to the public domain all rights
* for future licensing of the source code, all resale rights, and all
* publishing rights.
*
* We ask, but do not require, that the following message be included in
* all derived works:
*
* Portions developed at the National Center for Supercomputing
* Applications at the University of Illinois at Urbana-Champaign.
*
*
* THE UNIVERSITY OF ILLINOIS GIVES NO WARRANTY, EXPRESSED OR IMPLIED,
* FOR THE SOFTWARE AND/OR DOCUMENTATION PROVIDED, INCLUDING, WITHOUT
* LIMITATION, WARRANTY OF MERCHANTABILITY AND WARRANTY OF FITNESS FOR
* A PARTICULAR PURPOSE.
*
```

Figure F.1 README.FIRST (Continued)

```
developed by NCSA.  For more information about HDF, see the
January/February 1989 NCSA Data Link article, the document "NCSA HDF", and
the document "HDF  Specification".

This version of HDF runs on CRAYs running UNICOS, ALLIANTs, SUNs and
IRIS 4D machines running Unix, MACs running MacOS, VAXen running VMS and
PCs running MS/DOS.

Compilation of these programs produces a library of HDF routines that
can be called from either FORTRAN or C programs.

There is an older version of HDF implemented for CRAYs running CTSS, available
from NCSA.  This version only implements the 8-bit Raster Image Set.  If you
are interested in this, please contact Mike Folk (see below).


This document describes how to obtain a version of HDF for your system.
Other information, including hints on using HDF, descriptions of  the
files that comprise the distribution, and instructions on how to to
create a library, can be found in other README files in this directory.

There are two ways of obtaining HDF, depending on whether you are
accessing this system by remote login or anonymous ftp.
Accordingly, this document contains the following sections:

        Obtaining HDF using anonymous ftp
        Obtaining HDF using remote login

If you have any questions, problems or suggestions, you can contact us
via Email at mfolk@ncsa.uiuc.edu or likkai@ncsa.uiuc.edu,  or by writing
to Mike Folk, Software Development, NCSA, 605 East Springfield Ave.,
Champaign, IL 61820, or call 217 244 0647.


**************************************************************************
        Obtaining HDF using anonymous ftp
**************************************************************************

Login to ftp.ncsa.uiuc.edu (128.174.20.50), with a login name of
"anonymous". Give your real name as password.  Move to the directory
"HDF" by issuing the command "cd HDF" to ftp.  Now you are ready to
transfer files.  There are two ways to do this:

1. You may use the command "get hdf3.00.tar.Z" to download a compressed
"tar" format file.  (Be sure to set file transfer mode to binary with the
command "binary".)   Unpacking hdf.tar with the Unix "uncompress" and "tar"
utility on your system will produce a tree of subdirectories similar to
the ones in this directory.    These files are described in INSTALL.  They
must be compiled according to the instructions in that section.  (NOTE: this
tar file is very large, as it contains all the source files, plus
all of the documentation.  If space is dear, consider using the method
described in the next paragraph.)

1a) For MacII/MPW users, there is a binhexed stuffit file called
hdf3.00.sit.hqx.  Use ascii mode to get this file, unbinhex it, then
unstuff it.  This will provide you will all the files for the MacII.

2. As an alternative to "tar", you may download the files you require
directly.  Use "cd src" to move to the directory containing source
files.  Then use the command "mget *". If your system is VMS, get the
```

Figure F.1 README.FIRST (Continued)

Appendix F NCSA HDF README Files on Anonymous FTP

This appendix includes listings of the README files which can be found in the anonymous FTP directory that contains NCSA HDF. These listings were made on July 5, 1990 and do not necessarily reflect the current contents of the files. The best way to obtain the most recent versions is to access them through anonymous FTP.

Figure F.1. README.FIRST

```
#*****************************************************************************
#
#                          NCSA HDF version 3.1
#                                July 1, 1990
#
# NCSA HDF Version 3.1 source code and documentation are in the public
# domain.  Specifically, we give to the public domain all rights for future
# licensing of the source code, all resale rights, and all publishing rights.
#
# We ask, but do not require, that the following message be included in all
# derived works:
#
# Portions developed at the National Center for Supercomputing Applications at
# the University of Illinois at Urbana-Champaign.
#
# THE UNIVERSITY OF ILLINOIS GIVES NO WARRANTY, EXPRESSED OR IMPLIED, FOR THE
# SOFTWARE AND/OR DOCUMENTATION PROVIDED, INCLUDING, WITHOUT LIMITATION,
# WARRANTY OF MERCHANTABILITY AND WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE
#
#*****************************************************************************

*********************************************************************

                        NCSA HDF version 3.1
                           July 1, 1990
*********************************************************************

This is NCSA HDF version 3.1.  Suggestions and bug reports are welcome.

Included in this version are:
        the basic low-level routines to perform I/O to HDF files,
        routines to process 8-bit Raster Image Sets
        routines to process Scientific Data Sets.
        routines to process 24-bit Raster Image Sets
        routines to extract slabs from Scientific Data Sets
        routines to process Palettes (independently of images)
        routines to process Annotations for data items

NCSA HDF is the Hierarchical Data Format, a standard file format
```

Figure F.1 README.FIRST (Continued)

a data element.

DFupdate writes out the DD blocks necessary to update the file.

DFdup generates a DD whose offset and length are the same as those of another tag/ref.

DFdel deletes a tag/ref From the list of DDs.

DFerrno reports the value of DFerror.

DFishdf tells if a file is an HDF file

DFnewref generates an unused reference number

DFstat provides status information about an HDF file.

Utility Routines

hdfls displays the tags, ref numbers, and (optionally) lengths of data elements.

hdfed lets you browse in an HDF file and manipulate some of the data.

fptohdf converts floating-point data to HDF floating point data and/or 8-bit raster images.

r8tohdf converts one or more raw 8-bit images to HDF RIS8 format and writes them to a file, possibly with palettes.

hdftor8 converts images and or palettes from HDF format to raw format and stores them in two corresponding sets of files.

r24tohdf converts a raw RGB 24-bit image to an 8-bit RIS8 with a palette.

paltohdf converts a raw palette to hdf format.

hdftopal converts palette in an hdf file to raw format.

hdfseq/hdfrseq displays sequences of images directly to the screen from HDF files containing raster images.

DFSDputdata	**dspdata**	writes the data to the file, overwriting other file contents.
DFSDputslice	**dspslc**	writes part of a dataset to a file.
DFSDrestart	**dsfirst**	sets the next `get` command to read from the first SDS in the file, rather than the next.
DFSDsetdatastrs	**dissdast**	sets label, unit, and format specifications for the data.
DFSDsetdims	**dssdims**	sets the default rank and dimension sizes for succeeding files.
DFSDsetdimscale	**dssdisc**	sets the scale for a dimension.
DFSDsetdimstrs	**dssdist**	sets label, unit, and format specifications for a dimension and its scale.
DFSDsetlengths	**dsslens**	sets maximum lengths for strings that will hold labels, units, formats, and the name of the coordinate system.
DFSDsetmaxmin	**dssmaxm**	sets maximum and minimum data values.
DFSDsettype	**dsstype**	specifies data attributes—data type and representation, system type, and array order.
DFSDstartslice	**dsslc**	prepares system to write part of a dataset to a file.

General Purpose Routines

DFopen		provides an access path to the file named in filename with the access given in access.
DFclose		updates the DD blocks, then closes the access path to the file referred to by dfile.
DFsetfind	**dfsfind**	initializes searches for elements using tags or reference numbers.
DFfind	**dffind**	locates the data descriptor needed for the next read from the file.
DFgetelement	**dfget**	extracts the data referred to by the tag/ref and places the data in the array storage.
DFputelement	**dfput**	adds or replaces elements in dfile.
DFaccess		inititiates a read or write on the data element with the specified tag/ref combination.
DFread		reads a portion of a data element.
DFwrite		appends data to a data element.
DFseek		sets the read pointer to an offset within

DFPreadref	**dprref**	sets the reference number of the next palette to be retrieved.
DFPrestart	**dprest**	specifies that the next call to `DFPgetpal` reads the first palette in the file, rather than the next.
DFPwriteref	**dpwref**	sets the reference number of the next palette to be written.

Annotation Routines

DFANgetdesc	**dagdesc**	gets description of tag/ref.
DFANgetdesclen	**dagdlen**	gets length of description of tag/ref.
DFANgetlabel	**daglab**	gets label of tag/ref..
DFANgetlablen	**dagllen**	gets length of label of tag/ref.
DFANlablist	**dallist**	gets list of labels for a particular tag.
DFANlastref		returns ref of last annotation read or written.
DFANputdesc	**dapdesc**	puts description of tag/ref.
DFANputlabel	**daplab**	puts label of tag/ref.

Scientific Dataset Routines

DFSDadddata	**dsadata**	appends the data to the file, not overwriting other file contents.
DFSDclear	**dsclear**	clears all possible set values.
DFSDendslice	**dseslc**	indicates write completion for part of a dataset.
DFSDgetdata	**dsgdata**	reads the next dataset in the file.
DFSDgetdatastrs	**dsgdast**	reads the label, unit, and format specification for the data.
DFSDgetdims	**dsgdims**	gets the number of dimensions of the dataset and the sizes of the dimensions for the next SDS in the file.
DFSDgetdimscale	**dsgdisc**	reads the scale for a dimension.
DFSDgetdimstrs	**dsgdist**	reads the label, unit, and format for a dimension and its scale.
DFSDgetmaxmin	**dsgmaxm**	reads the maximum and minimum values.
DFSDgetslice	**dsgslc**	reads part of a dataset.

Appendix E Routine Lists

Raster Image Routines

DFR8addimage	**d8aimg**	appends the RIS8 for the image to the file.
DFR8getdims	**d8gdims**	retrieves the dimensions of the image and indicates whether a palette is associated and stored with the image.
DFR8getimage	**d8gimg**	retrieves the image and any associated palette, and stores them in arrays.
DFR8putimage	**d8pimg**	writes out the RIS8 for the image as the first image in the file.
DFR8restart	**d8first**	sets the next `get` command to read from the first RIS8 in the file, rather than the next.
DFR8setpalette	**d8spal**	sets the default palette to be used for subsequent images.
DF24addimage	**d2iaimg**	appends the RIS24 for the image to the file.
DF24getdims	**d2igdims**	retrieves the dimensions and interlace of the image.
DF24getimage	**d2igimg**	retrieves the image and stores it in an array.
DF24reqil	**d2reqil**	specifies an interlace to be used in place of the interlace indicated in the file when the next raster image is read.
DF24setil	**d2setil**	sets the interlace to be used when writing out the RIS24 for the image.

Palette Routines

DFPaddpal	**dpapal**	appends a palette to a file.
DFPgetpal	**dpgpal**	reads in the next palette in the file.
DFPlastref	**dplref**	returns the value of the reference number most recently read or written.
DFPnpals	**dpnpals**	indicates number of palettes in a file.
DFPputpal	**dpppal**	writes a palette to a file.

FORTRAN:

```
cf77 myprog.f –o myprog –ldf
```

where `myprog` is the name of the executable program.

As another example, suppose you have a program called `myprog.c` written in C for one of the Sun systems. If `myprog.c` contains calls to HDF routines, you can link `libdf.a` to your program when you compile it by entering:

C:

```
cc  myprog.c –o myprog /soft/hdf/lib/libdf.a
```

Appendix D Public HDF Directories on NCSA Computers

There are public HDF directories on several machines at NCSA. In each supported HDF directory you will find the following:

- A README file that gives further information about the software and how to use it (Be sure to read this file before using the software, because it may contain important information about recent changes to the software.)
- The subdirectory `lib/` with the library file `libdf.a`, which contains the high-level routines described in this manual for working with raster image sets and scientific datasets, as well as the lower-level general purpose routines for building and manipulating HDF routines of any type
- The subdirectory `bin/` with the executable utility programs
- The subdirectory `src/`, which contains the source code for the latest supported version of all programs
- The subdirectory `include/`, which contains the header files listed in Appendix B of this manual
- The subdirectory `examples/`, which contains one or more sample programs that use HDF

The HDF public directories are currently accessible on the CRAY-2, CRAY X-MP, Alliant FX/8 (medusa), Alliant FX/80 (replicant), and NCSA Sun systems. The pathnames of these directories are listed in Table D.1.

Table D.1 **Pathnames of NCSA HDF Directories**

NCSA Computer	Directory Path
CRAY-2	`/usr/lib`
CRAY X-MP	`/usr/lib`
Alliant FX/80 (replicant)	`/usr/hdf/lib`
Sun systems	`/soft/hdf/lib`
SGI systems	`/rels/shared/soft/hdf`

In order to compile a program that uses one of the NCSA HDF library routines, you need to link the library to your program when you compile or link your program. For example, suppose you have a program called `myprog.f` written in FORTRAN for the CRAY-2 system. If `myprog.f` contains calls to HDF routines, you can link `libdf.a` to your program when you compile it by entering:

Table C.1 **Long and Short Version FORTRAN Names (Continued)**

Long Version	Short Version
DFSDadddata	dsadata
DFSDclear	dsclear
DFSDendslice	dseslc
DFSDgetdata	dsgdata
DFSDgetdatalen	dsgdaln
DFSDgetdatastrs	dsgdast
DFSDgetdimlen	dsgdiln
DFSDgetdims	dsgdims
DFSDgetdimscale	dsgdisc
DFSDgetdimstrs	dsgdist
DFSDgetmaxmin	dsgmaxm
DFSDgetslice	dsgslc
DFSDputdata	dspdata
DFSDputslice	dspslc
DFSDrestart	dsfirst
DFSDsetdatastrs	dssdast
DFSDsetdims	dssdims
DFSDsetdimscale	dssdisc
DFSDsetdimstrs	dssdist
DFSDsetlengths	dsslens
DFSDsetmaxmin	dssmaxm
DFSDsettype	dsstype
DFSDstartslice	dssslc

Appendix C Eight-Character FORTRAN Names

Some FORTRAN compilers on machines that NCSA supports, such as UNICOS FORTRAN (CFT77) on the CRAY-2 system, only accept identifier names that are eight or fewer characters. Therefore, a set of equivalent names has been devised that can be used when programming with one of these compilers. Table C.1 contains official names, together with the shorter, less descriptive versions. Both sets of names are supported on all HDF-supported machines.

Table C.1 **Long and Short Version FORTRAN Names**

Long Version	Short Version
DF24addimage	d2iaimg
DF24getdims	d2igdims
DF24getimage	d2igimg
DF24reqil(il)	d2reqil
DF24setil	d2setil
DFANgetdesc	dagdesc
DFANgetdesclen	dagdlen
DFANgetlabel	daglab
DFANgetlablen	dagllen
DFANlablist	dallist
DFANlastref	————
DFANputdesc	dapdesc
DFANputlabel	daplab
DFPaddpal	dpaPal
DFPgetpal	dpgpal
DFPlastref	dplref
DFPnpals	dpnpals
DFPputpal	dpppal
DFPreadref	dprref
DFPrestart	dprest
DFPwriteref	dpwref
DFR8addimage	d8aimg
DFR8getdims	d8gdims
DFR8getimage	d8gimg
DFR8putimage	d8pimg
DFR8restart	d8first
DFR8setpalette	d8spal

```
     *           DFE_NODIM     = -34,
     *           DFE_NOTENOUGH       = -35)

C                       Logical Constants
      PARAMETER (DFACC_READ    = 1,
     *           DFACC_WRITE   = 2,
     *           DFACC_CREATE  = 4,
     *           DFACC_ALL     = 7)
```

```
      *
      *           DFTAG_RIG     = 306,
      *           DFTAG_LD      = 307,
      *           DFTAG_MD      = 308,
      *           DFTAG_MA      = 309,
```

Figure B.3 FORTRAN: Constants File constants.h (Continued)

```
       PARAMETER (DFTAG_CCN     = 310,
      *           DFTAG_CFM     = 311,
      *           DFTAG_AR      = 312,
      *           DFTAG_DRAW    = 400,
      *           DFTAG_RUN     = 401,
      *
      *           DFTAG_XYP     = 500,
      *           DFTAG_MTO     = 501,
      *
      *           DFTAG_T14     = 602,
      *           DFTAG_T105    = 603,
      *
      *           DFTAG_RLE     = 11,
      *           DFTAG_IMCOMP  = 12)

C                               Error Return Codes
       PARAMETER (DFE_NOERROR    =  0,
      *           DFE_FNF        = -1,
      *           DFE_DENIED     = -2,
      *           DFE_ALROPEN    = -3,
      *           DFE_TOOMANY    = -4,
      *           DFE_BADNAME    = -5,
      *           DFE_BADACC     = -6 ,
      *           DFE_NOTOPEN    = -8,
      *           DFE_CANTCLOSE        = -9,
      *           DFE_DFNULL    = -10,
      *           DFE_ILLTYPE   = -11,
      *           DFE_UNSUPPORTED     = -12,
      *           DFE_BADDDLIST       = -13,
      *           DFE_NOTDFFILE       = -14,
      *           DFE_SEEDTWICE       = -15,
      *           DFE_NOSPACE   = -16,
      *           DFE_READERROR       = -18,
      *           DFE_WRITEERROR      = -19)

       PARAMETER (DFE_SEEKERROR       = -20,
      *           DFE_NOFREEDD  = -21,
      *           DFE_BADTAG    = -22,
      *           DFE_BADREF    = -23,
      *           DFE_RDONLY    = -24,
      *           DFE_BADCALL   = -25,
      *           DFE_BADPTR    = -26,
      *           DFE_BADLEN    = -27,
      *           DFE_BADSEEK   = -28,
      *           DFE_NOMATCH   = -29,
      *           DFE_NOTINSET  = -30,
      *           DFE_BADDIM    = -31,
      *           DFE_BADOFFSET       = -32,
      *           DFE_BADSCHEME       = -33,
```

```
#endif DFTAG_NULL
```

Figure B.3 FORTRAN: Constants File constants.h

```
/****************************************************************************
*
*                                                    NCSA HDF version 2.0
*                                                         December 20, 1988
*
* NCSA HDF Version 2.0 source code and documentation are in the public domain.
* Specifically, we give to the public domain all rights for future licensing
* of the source code, all resale rights, and all publishing rights.
*
* We ask, but do not require, that the following message be included in all
* derived works:
*
* Portions developed at the National Center for Supercomputing Applications at
* the University of Illinois at Urbana-Champaign.
*
* THE UNIVERSITY OF ILLINOIS GIVES NO WARRANTY, EXPRESSED OR IMPLIED, FOR THE
* SOFTWARE AND/OR DOCUMENTATION PROVIDED, INCLUDING, WITHOUT LIMITATION,
* WARRANTY OF MERCHANTABILITY AND WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE
*
****************************************************************************/
C *----------------------------------------
C * File:     dfF.h
C * Purpose:  Fortran header file for HDF routines
C * Contents:
C *    Tag definitions
C * Error return codes
C *    Logical constants
C * Remarks: This file can be included with user programs
C *----------------------------------------*

       PARAMETER (DFREF_WILDCARD = 0,
      *           DFTAG_WILDCARD = 0,
      *           DFTAG_NULL     = 1)

       PARAMETER (DFTAG_FID     = 100,
      *           DFTAG_FD      = 101,
      *           DFTAG_TID     = 102,
      *           DFTAG_TD      = 103,
      *           DFTAG_DIL     = 104,
      *           DFTAG_DIA     = 105,
      *           DFTAG_NT      = 106,
      *           DFTAG_MT      = 107,
      *
      *           DFTAG_ID8     = 200,
      *           DFTAG_IP8     = 201,
      *           DFTAG_RI8     = 202,
      *           DFTAG_CI8     = 203,
      *           DFTAG_II8     = 204)
      *
       PARAMETER (DFTAG_ID      = 300,
      *           DFTAG_LUT     = 301,
      *           DFTAG_RI      = 302,
      *           DFTAG_CI      = 303,
```

```
#define DFTAG_SDC      708            /* Coord sys */

       /* compression schemes */
#define DFTAG_RLE      11             /* run length encoding */
#define DFTAG_IMCOMP 12               /* IMCOMP compression */

/*————————————————————————————*/
/*             Error Return Codes                                    */

#define DFE_NOERROR           0       /* No error */
#define DFE_FNF        -1     /* File not found error */
#define DFE_DENIED            -2      /* Access to file denied */
#define DFE_ALROPEN           -3      /* File already open */
#define DFE_TOOMANY           -4      /* Too Many DF's or files open */
#define DFE_BADNAME           -5      /* Bad file name on open */
#define DFE_BADACC            -6      /* Bad file access mode */
#define DFE_BADOPEN           -7      /* Other open error */
#define DFE_NOTOPEN           -8      /* File can't be closed: it isn't open */
#define DFE_CANTCLOSE         -9      /* fclose wouldn't work! */
#define DFE_DFNULL            -10     /* DF is a null pointer */
#define DFE_ILLTYPE           -11     /* DF has an illegal type: internal error */
#define DFE_UNSUPPORTED       -12     /* Feature not currently supported */
#define DFE_BADDDLIST         -13     /* No DD list: internal error */
#define DFE_NOTDFFILE         -14     /* Not a DF file and it is not 0 length */
#define DFE_SEEDTWICE         -15     /* DD list already seeded: internal error */
#define DFE_NOSPACE           -16     /* Malloc failed */
#define DFE_NOSUCHTAG         -17     /* No such tag in the file: search failed*/
#define DFE_READERROR         -18     /* There was a read error */
#define DFE_WRITEERROR -19    /* There was a write error */
#define DFE_SEEKERROR         -20     /* There was a seek error */
#define DFE_NOFREEDD          -21     /* No free DD's left: internal error */
#define DFE_BADTAG            -22     /* illegal WILDCARD tag */
#define DFE_BADREF            -23     /* illegal WILDCARD reference # */
#define DFE_RDONLY            -24     /* The DF is read only */
#define DFE_BADCALL           -25     /* Calls in wrong order */
#define DFE_BADPTR            -26     /* NULL ptr argument */
#define DFE_BADLEN            -27     /* negative len specified */
#define DFE_BADSEEK           -28     /* Attempt to seek past end of element */
#define DFE_NOMATCH           -29     /* No (more) DDs which match specified tag/ref */
#define DFE_NOTINSET          -30     /* Warning: Set contained unknown tag: ignored */
#define DFE_BADDIM            -31     /* negative or zero dimensions specified */
#define DFE_BADOFFSET         -32     /* Illegal offset specified */
#define DFE_BADSCHEME         -33     /* Unknown compression scheme specified */
#define DFE_NODIM             -34     /* No dimension record associated with image */
#define DFE_NOTENOUGH         -35     /* space provided insufficient for size of data
```

Figure B.2 C Header File: df.h (Continued)

```
*/
#define DFE_NOVALS            -36     /* Values not available */
#define DFE_CORRUPT           -37     /* File is corrupted */
#define DFE_BADCONV           -37     /* Don't know how to convert data type */
#define DFE_BADFP             -38     /* The file contained an illegal floating point no*/

/*————————————————————————————*/
/*             Logical Constants                                     */

#define DFACC_READ            1       /* Read Access */
#define DFACC_WRITE           2       /* Write Access */
#define DFACC_CREATE          4       /* force file to be created */
#define DFACC_ALL             7       /* logical and of all the above values */
```

```
        DFerror;                        /* Error code for DF routines */

/*————————————————————————————————————*/
/*              Tag Definitions                                          */

#define DFREF_WILDCARD 0        /* wildcard ref for searches */

#define DFTAG_WILDCARD 0        /* wildcard tag for searches */
#define DFTAG_NULL     1                /* empty DD */

        /* utility set */
#define DFTAG_FID      100              /* File identifier */
#define DFTAG_FD       101              /* File description */
#define DFTAG_TID      102              /* Tag identifier */
#define DFTAG_TD       103              /* Tag descriptor */
#define DFTAG_DIL      104              /* data identifier label */
#define DFTAG_DIA      105              /* data identifier annotation */
#define DFTAG_NT       106              /* number type */
#define DFTAG_MT       107              /* machine type */

        /* raster-8 set */
#define DFTAG_ID8      200              /* 8-bit Image dimension */
#define DFTAG_IP8      201              /* 8-bit Image palette */
#define DFTAG_RI8      202              /* Raster-8 image */
#define DFTAG_CI8      203              /* RLE compressed 8-bit image */
#define DFTAG_II8      204              /* IMCOMP compressed 8-bit image */

        /* Raster Image set */
#define DFTAG_ID       300              /* Image DimRec */
#define DFTAG_LUT      301              /* Image Palette */
#define DFTAG_RI       302              /* Raster Image */
#define DFTAG_CI       303              /* Compressed Image */

#define DFTAG_RIG      306              /* Raster Image Group */
#define DFTAG_LD       307              /* Palette DimRec */
#define DFTAG_MD       308              /* Matte DimRec */
#define DFTAG_MA       309              /* Matte Data */
#define DFTAG_CCN      310              /* color correction */
#define DFTAG_CFM      311              /* color format */
#define DFTAG_AR       312              /* aspect ratio */

#define DFTAG_DRAW     400              /* Draw these images in sequence */
#define DFTAG_RUN      401              /* run this as a program/script */
```

Figure B.2 C Header File: df.h (Continued)

```
#define DFTAG_XYP      500              /* x-y position */
#define DFTAG_MTO      501              /* machine-type override */

        /* Tektronix */
#define DFTAG_T14      602              /* TEK 4014 data */
#define DFTAG_T105     603              /* TEK 4105 data */

        /* Scientific Data set */
#define DFTAG_SDG      700              /* Scientific Data Group */
#define DFTAG_SDD      701              /* Scientific Data DimRec */
#define DFTAG_SD       702              /* Scientific Data */
#define DFTAG_SDS      703              /* Scales */
#define DFTAG_SDL      704              /* Labels */
#define DFTAG_SDU      705              /* Units */
#define DFTAG_SDF      706              /* Formats */
#define DFTAG_SDM      707              /* Max/Min */
```

```
        /* DF is the internal structure associated with each DF file */
        /* It holds information associated with the file as a whole */
        /* ### Note: there are hooks for having multiple DF files open at a time */
typedef struct DF {
        DFdle
                *list,          /* Pointer to the DLE list */
                *last_dle;      /* last_dle and last_dd are used in searches to indicate
                                        element returned by previous call to DFfind */
        int
                type,           /* 0= not in use, 1= normal, -1 = multiple */
                                /* this is a hook for when multiple files are open */
                access,/* permitted access types: 0=none, 1=r, 2=w, 3=r/w */
                changed,        /* True if anything in DDs modified since last write */
                last_tag,       /* Last tag searched for by DFfind */
                last_ref,       /* Last reference number searched for */
                last_dd,        /* see last_dle */
                defdds,/* default number of DD's in each block */
                up_access;      /* access permissions to element being read/updated */
                                /* Used by DFstart */
        DFdd
                *up_dd;/* DD of element being read/updated, used by DFstart */
                                /* file handle is a file pointer or file descriptor */
                                /* depending whether we use buffered or unbuffered i/o */

#ifdef DF_BUFFIO
        FILE *                  /* file pointer */
#else DF_BUFFIO
        int                     /* file descriptor */
#endif DF_BUFFIO
                file;           /* File handle for real file */
} DF;


typedef struct DFdi { /* data identifier: specifies data element uniquely */
        uint16 tag;
        uint16 ref;
} DFdi;



typedef struct DFdata {       /* structure for returning status information */
        int version;          /* version number of program */
} DFdata;
```

Figure B.2 C Header File: df.h (Continued)

```
/*————————————————————————————————————*/
/*              Procedure types                                         */

DF *DFopen();
int32 DFgetelement();
int32 DFread();
int32 DFseek();
int32 DFwrite();

/*————————————————————————————————————*/
/*              Global Variables                                        */

#ifndef DFMASTER
extern
#endif DFMASTER
int
```

```
* WARRANTY OF MERCHANTABILITY AND WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE
*
*****************************************************************************/

/*----------------------------------------
 * File:       df.h
 * Purpose:    header file for HDF routines
 * Invokes:    dfi.h
 * Contents:
 *      Structure definitions: DFddh, DFdd, DFdesc, DFdle, DF, DFdi, DFdata
 *      Procedure type definitions
 *      Global variables
 *      Tag definitions
 *  Error return codes
 *      Logical constants
 * Remarks: This file is included with user programs
 *             Since it includes stdio.h etc., do not include these after df.h
 *----------------------------------------*/


#ifndef DFTAG_NULL                          /* avoid re-inclusion */

/* include DF (internal) header information */
#include "dfi.h"

/*----------------------------------------*/
/*Type declarations                                                   */

typedef struct DFddh {              /* format of data descriptor headers as in file */
       int16
              dds;                  /* number of dds in header block */
       int32
              next;                 /* offset of next header block */
       } DFddh;


typedef struct
       uint16
              tag,                  /* data tag */
              ref;                  /* data reference number */
       int32
              offset,        /* offset of data element in file */
              length;        /* number of bytes */
       } DFdd;
```

Figure B.2 C Header File: df.h (Continued)

```
       /* descriptor structure is same as dd structure.  ###Note: may be changed */
#define DFdesc  DFdd


       /* DLE is the internal structure that stores data descriptor information */
       /* It is a linked list of DDs */
typedef struct DFdle {              /* Data List element */
       struct DFdle
              *next;                /* link to next dle */
       DFddh
              ddh;                  /* To store headers */
       DFdd
              dd[1];                /* dummy size */
       } DFdle;
```

```
#define DFE_DENIED     -2
#define DFE_ALROPEN    -3
#define DFE_TOOMANY    -4
#define DFE_BADNAME    -5
#define DFE_BADACC     -6
#define DFE_NOTOPEN    -8
#define DFE_CANTCLOSE  -9
#define DFE_DFNULL     -10
#define DFE_ILLTYPE    -11
#define DFE_UNSUPPORTED       -12
#define DFE_BADDDLIST         -13
#define DFE_NOTDFFILE         -14
#define DFE_SEEDTWICE         -15
#define DFE_NOSPACE           -16
#define DFE_READERROR         -18
#define DFE_WRITEERROR -19
#define DFE_SEEKERROR         -20
#define DFE_NOFREEDD          -21
#define DFE_BADTAG            -22
#define DFE_BADREF            -23
#define DFE_RDONLY            -24
#define DFE_BADCALL           -25
#define DFE_BADPTR            -26
#define DFE_BADLEN            -27
#define DFE_BADSEEK           -28
#define DFE_NOMATCH           -29
#define DFE_NOTINSET          -30
#define DFE_BADDIM            -31
#define DFE_BADOFFSET         -32
#define DFE_BADSCHEME         -33
#define DFE_NODIM             -34
#define DFE_NOTENOUGH         -35

C                                       Logical Constants

#define DFACC_READ            1
#define DFACC_WRITE           2
#define DFACC_CREATE          4
#define DFACC_ALL             7

#endif DFTAG_NULL
```

Figure B.2 C Header File: df.h

```
/****************************************************************************
*
*                                                    NCSA HDF version 2.0
*                                                         December 20, 1988
*
* NCSA HDF Version 2.0 source code and documentation are in the public domain.
* Specifically, we give to the public domain all rights for future licensing
* of the source code, all resale rights, and all publishing rights.
*
* We ask, but do not require, that the following message be included in all
* derived works:
*
* Portions developed at the National Center for Supercomputing Applications at
* the University of Illinois at Urbana-Champaign.
*
*
* THE UNIVERSITY OF ILLINOIS GIVES NO WARRANTY, EXPRESSED OR IMPLIED, FOR THE
* SOFTWARE AND/OR DOCUMENTATION PROVIDED, INCLUDING, WITHOUT LIMITATION,
```

```
C * File:      dfF.h
C * Purpose:   Fortran header file for HDF routines
C * Contents:
C *     Tag definitions
C * Error return codes
C *     Logical constants
C * Remarks: This file can be included with user programs
C *-------------------------------------*

#ifndef DFTAG_NULL

#define DFREF_WILDCARD 0
#define DFTAG_WILDCARD 0
#define DFTAG_NULL     1

#define DFTAG_FID      100
#define DFTAG_FD       101
#define DFTAG_TID      102
#define DFTAG_TD       103
#define DFTAG_DIL      104
#define DFTAG_DIA      105
#define DFTAG_NT       106
#define DFTAG_MT       107

#define DFTAG_ID8      200
#define DFTAG_IP8      201
#define DFTAG_RI8      202
#define DFTAG_CI8      203
#define DFTAG_II8      204

#define DFTAG_ID       300
#define DFTAG_LUT      301
#define DFTAG_RI       302
#define DFTAG_CI       303

#define DFTAG_RIG      306
#define DFTAG_LD       307
#define DFTAG_MD       308
#define DFTAG_MA       309
```

Figure B.1 FORTRAN Header File: dfF.h (Continued)

```
#define DFTAG_CCN      310
#define DFTAG_CFM      311
#define DFTAG_AR       312

#define DFTAG_DRAW     400
#define DFTAG_RUN      401

#define DFTAG_XYP      500
#define DFTAG_MTO      501

#define DFTAG_T14      602
#define DFTAG_T105     603

#define DFTAG_RLE      11
#define DFTAG_IMCOMP 12

C                                  Error Return Codes

#define DFE_NOERROR    0
#define DFE_FNF -1
```

Appendix B Header Files and FORTRAN Constants File

This appendix includes the general header files used in compiling all HDF libraries. These files do *not* have to be included with most applications, but they could be usefully added to code that refers to specific tags by name, or code that is designed to be responsive to different error conditions. You get these files when you get the HDF source code.

You will see that the C header (`df.h`) shown in Figure B.2 is more completely commented than the FORTRAN header () shown in Figure B.1. Thus, you may want to look at the C header for explanations of some definitions, even if you plan to use the #define statements from the FORTRAN header.

The file `constants.f` shown in Figure B.3 is equivalent to `dfF.h` (Figure B.1), put uses the FORTRAN `PARAMETER` statement to declare constants, and hence may be more useful for inserting into your FORTRAN code.

Other headers used for specific applications, such as the RIS interface and the SDS interface can be found by obtaining the source code from NCSA. These headers also do not normally have to be included with your source code.

Figure B.1 FORTRAN Header File: dfF.h

```
/*****************************************************************************
*
*                                                    NCSA HDF version 2.0
*                                                         December 20, 1988
*
* NCSA HDF Version 2.0 source code and documentation are in the public domain.
* Specifically, we give to the public domain all rights for future licensing
* of the source code, all resale rights, and all publishing rights.
*
* We ask, but do not require, that the following message be included in all
* derived works:
*
* Portions developed at the National Center for Supercomputing Applications at
* the University of Illinois at Urbana-Champaign.
*
* THE UNIVERSITY OF ILLINOIS GIVES NO WARRANTY, EXPRESSED OR IMPLIED, FOR THE
* SOFTWARE AND/OR DOCUMENTATION PROVIDED, INCLUDING, WITHOUT LIMITATION,
* WARRANTY OF MERCHANTABILITY AND WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE
*
*****************************************************************************/

C *-------------------------------------
```

DFTAG_SD	Scientific Data	reals	702	points to scientific data.
DFTAG_SDS	Scales	reals	703	shows scales to be used when interpreting and displaying data.
DFTAG_SDL	Labels	strings	704	labels for all dimensions and for the data.
DFTAG_SDU	Units	strings	705	displays units for all dimensions and for the data.
DFTAG_SDF	Formats	strings	706	formats for displaying axes and data.
DFTAG_SDM	Max/min	2 reals	707	shows maximum and minimum values for the data.
DFTAG_SDC	Coordinate system	string	708	shows coordinate system to be used to interpret data.
DFTAG_SDT	Transpose	0 bytes	709	indicates that data is transposed in the file.

DFTAG_RI	Raster Image	*x*y* bytes	302	points to raster image data.
DFTAG_CI	Compressed Image	*n* bytes	303	points to a compressed raster image.
DFTAG_RIG	Raster Image Group	*n**4 bytes	306	lists the Data Identifiers (tag/ref) that describe a raster image set.
DFTAG_LD	LUT Dimension	20 bytes	307	defines the dimensions of the 2D array that its corresponding LUT tag refers to.
DFTAG_MD	Matte Dimension	20 bytes	308	defines the dimensions of the 2D array that its corresponding MA tag refers to.
DFTAG_MA	Matte Data	*n* bytes	309	points to matte data.
DFTAG_CCN	Color Correction	*n* bytes	310	specifies the Gamma correction for the image and color primaries for the generation of the image.
DFTAG_CFM	Color Format	string	311	indicates what interpretation should be given to each element of each pixel in a raster image.
DFTAG_AR	Aspect Ratio	4 bytes	312	indicates the visual aspect ratio for this image.

Table A.1 HDF Tags (Continued)

Tag	Name	Data Type	Number	Use
Composite Image Tag				
DFTAG_DRAW	Draw	*n**4 bytes	400	identifies a list of Data Identifiers (tag/ref) which define a composite image.
DFTAG_RUN	Run	*n* bytes	401	identifies code that is to be executed as a program or script.
Tag for Composite or Raster Images				
DFTAG_XYP	XY Position	8 bytes	500	indicates an XY position on the screen for composites and other groups.
Vector Image Tags				
DFTAG_T14	Tektronix 4014	*n* bytes	602	points to a Tek 4014 data stream. The bytes in the data field, when read and sent to a Tektronix 4014 terminal, will be displayed as a vector image.
DFTAG_T105	Tektronix 4105	*n* bytes	603	points to a Tek 4105 data stream. The bytes in the data field, when read and sent to a Tektronix 4105 terminal, will be displayed as a vector image.
Scientific Data Set Tags				
DFTAG_SDG	Scientific Data Group	*n**4 bytes	700	lists the Data Identifiers (tag/ref) that describe a scientific dataset.
DFTAG_SDD	Scientific Data Dimension Record	*n* bytes	701	defines the rank and dimensions of the array that the corresponding SD refers to.

DFTAG_FD	File Descriptor	text	101	points to a block of text describing the overall file contents. It is intended to be user-supplied comments about the file.
DFTAG_TID	Tag Identifier	string	102	provides a way for a human reader to tell the meaning of a tag stored in a file.
DFTAG_TD	Tag Descriptor	text	103	similar to TID, but allows more extensive text to be included.
DFTAG_DIL	Data Identifier Label	string	104	associates the string with the Data Identifier as a label for whatever that Data Identifier points to in turn. By including DILs, any piece of data can be given a label for future reference. For example, this is often used to give titles to images.
DFTAG_DIA	Data Identifier Annotation	text	105	associates the text block with the Data Identifier as annotation for whatever that Data Identifier points to in turn. With DIA, any Data Identifier can have a lengthy, user-written description of why that data is in the file.
DFTAG_NT	Number Type	4 bytes	106	used by any other element in the file to indicate specifically what a numeric value looks like.
DFTAG_MT	Machine Type	0 bytes	107	specifies that all unconstrained or partially constrained values in this HDF file are of the default type for that hardware.

Table A.1 HDF Tags (Continued)

Tag	Name	Data Type	Number	Use

Raster-8 (8-Bit Only) Tags

DFTAG_ID8	Image Dimension-8	4 bytes	200	two 16-bit integers representing the width and height of an 8-bit raster image in bytes.
DFTAG_IP8	Image Palette-8	768 bytes	201	a 256 x 3 byte array representing the red, green and blue elements of the 256-color palette respectively.
DFTAG_RI8	Raster Image-8	*x*y* bytes	202	a row-wise representation of the elementary 8-bit image data.
DFTAG_CI8	Compressed Image-8	*n* bytes	203	a row-wise representation of the elementary 8-bit image data. Each row is compressed using a form of run-length encoding.
DFTAG_II8	IMCOMP Image-8	*n* bytes	204	a 4:1 compressed 8-bit image, using the IMCOMP compression scheme.

General Raster Image Tags

DFTAG_ID	Image Dimension	20 bytes	300	defines the dimensions of the 2D array that its corresponding RI tag refers to.
DFTAG_LUT	Look Up Table	*n* bytes	301	table to be used by hardware to assign RGB colors or HSV colors to data values.

Appendix A NCSA HDF Tags

Overview

This appendix includes a Table containing brief descriptions of most of the tags that have been assigned at NCSA for general use. The list is growing and will be expanded in future editions of the documentation. More detailed descriptions of the tags can be found in an appendix of the *NCSA HDF Specifications*. To obtain a draft of this document, contact the NCSA Software Development Group at the address listed on the README page at the beginning of this manual.

Tag Types and Descriptions

The following table has five columns. The *Tag column* gives the abbreviated symbolic name of the tag that is often used in an augmented form in HDF programs. The *Name column* gives a name commonly used in human-readable media. The *Data Type column* describes the type of data that is associated with the tag, and where possible, gives the amount of data. The *Number column* gives the actual numeric value that is stored in the file to represent the tag. The *Use column* describes the tag in brief and general terms.

In the table, the term *string* refers to a sequence of ASCII characters, with a zero byte possibly occurring at the end, but nowhere else. The term *text* likewise refers to a sequence of ASCII characters, but it may contain zero bytes elsewhere than at the end.

Table A.1 HDF Tags

Tag	Name	Datatype	Number	Use
Utility Tags				
DFTAG_NULL	No Data	none	001	used for place holding and filling up empty portions of the Data Descriptor Block.
DFTAG_RLE	Run length encoding	0 bytes	011	specifies that run length encoding is used to compress an image.
DFTAG_IMC	IMCOMP Compression	0 bytes	012	specifies that IMCOMP compression is used to compress an image.
DFTAG_FID	File Identifier	string	100	points to a string that the user wants to associate with this file. It is intended to be a user-supplied title for the file.