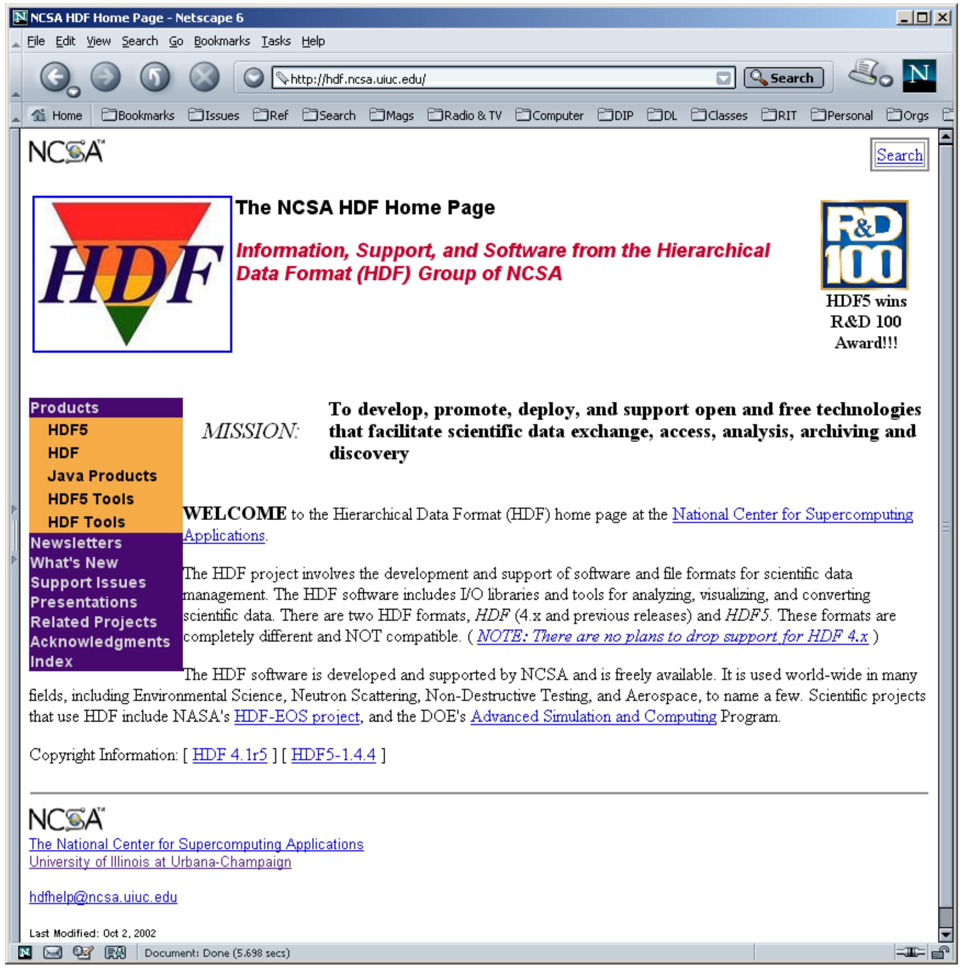
Hierarchical Data Format

SIMG-726

January 2003

http://hdf.ncsa.uiuc.edu/


NCSA HDF Home Page - Netscape 6
File Edit View Search Go Bookmarks Tasks Help
http://hdf.ncsa.uiuc.edu/
Search
Home Bookmarks Issues Ref Search Mags Radio & TV Computer DIP DL Classes RIT Personal Orgs
NCSA
Search
HDF
R&D 100
Products
HDF5
HDF
Java Products
HDF5 Tools
HDF Tools
Newsletters
What's New
Support Issues
Presentations
Related Projects
Acknowledgments
Index
Document: Done (5.698 secs)

The NCSA HDF Home Page

Information, Support, and Software from the Hierarchical Data Format (HDF) Group of NCSA

HDF5 wins R&D 100 Award!!!

MISSION: **To develop, promote, deploy, and support open and free technologies that facilitate scientific data exchange, access, analysis, archiving and discovery**

WELCOME to the Hierarchical Data Format (HDF) home page at the National Center for Supercomputing Applications.

The HDF project involves the development and support of software and file formats for scientific data management. The HDF software includes I/O libraries and tools for analyzing, visualizing, and converting scientific data. There are two HDF formats, *HDF* (4.x and previous releases) and *HDF5*. These formats are completely different and NOT compatible. (*NOTE: There are no plans to drop support for HDF 4.x*)

The HDF software is developed and supported by NCSA and is freely available. It is used world-wide in many fields, including Environmental Science, Neutron Scattering, Non-Destructive Testing, and Aerospace, to name a few. Scientific projects that use HDF include NASA's HDF-EOS project, and the DOE's Advanced Simulation and Computing Program.

Copyright Information: [HDF 4.1r5] [HDF5-1.4.4]

NCSA
The National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign

hdfhelp@ncsa.uiuc.edu

Last Modified: Oct 2, 2002

ftp://ftp.ncsa.uiuc.edu/HDF/HDF/Documentation/HDF4.1r5/Users_Guide/

What is HDF?

The Hierarchical Data Format, or HDF, is a multiobject file format for sharing scientific data in a distributed environment.

HDF was created at the National Center for Supercomputing Applications to serve the needs of diverse groups of scientists working on projects in various fields.

HDF was designed to address many requirements for storing scientific data, including:

- Support for the types of data and metadata commonly used by scientists.
- Efficient storage of and access to large data sets.
- Platform independence.
- Extensibility for future enhancements and compatibility with other standard formats.

Metadata

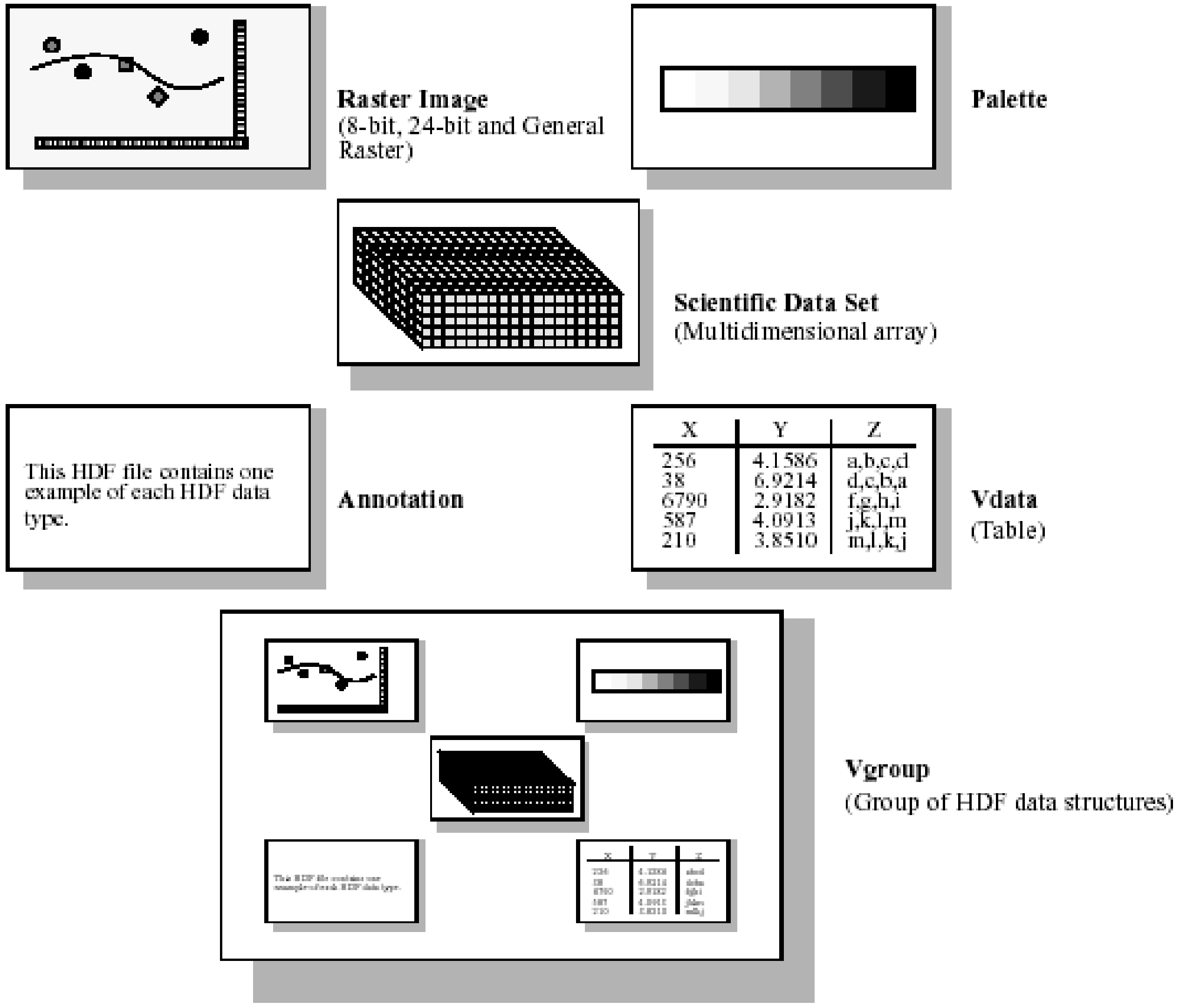
- HDF files are self-describing: for each HDF data structure in a file, there is comprehensive information about the data and its location in the file.

- Many types of data can be included within an HDF file.
  - Symbolic
  - Numerical
  - Graphical

Why was HDF Created?

- Scientific data is generated and processed on several platforms using various software packages.
- There may be several kinds of information to include in one file or file group.
- Information from one application may vary from one file to another.
- Files may be related conceptually but separated physically.
- Data needs to be shared among researchers, who have different uses for it.

HDF Data Structures


Raster Image
(8-bit, 24-bit and General Raster)
Palette
Scientific Data Set
(Multidimensional array)
This HDF file contains one example of each HDF data type.
Annotation
Vdata
(Table)
Vgroup
(Group of HDF data structures)

X	Y	Z
256	4.1586	a,b,c,d
38	6.9214	d,c,b,a
6790	2.9182	f,g,h,i
587	4.0913	j,k,l,m
210	3.8510	m,l,k,j

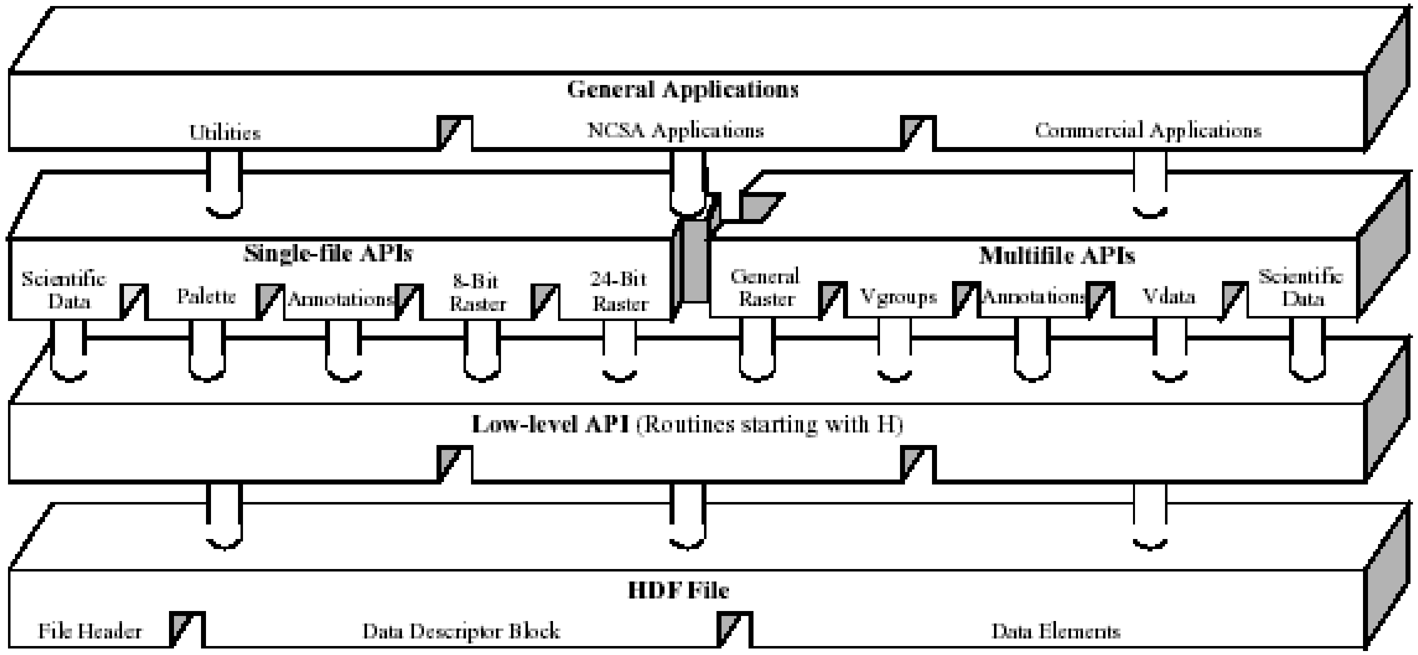
High-Level Functionality

- Provides the mechanism for programs to obtain information about the data in a file from within the file, rather than from another source.
- Lets the user store mixtures of data from different sources into a single file as well as store the data and its related information in separate files, even when the files are processed by the same application program.
- Standardizes the formats and descriptions of many types of commonly-used data sets, such as raster images and multidimensional arrays.
- Encourages the use of a common data format by all machines and programs that produce files containing specific data.
- Can be adapted to accommodate virtually any kind of data.

HDF Interface Structure

HDF APIs are divided into two categories: multifile interfaces (new) and single-file interfaces (old). The multifile interfaces are those that provide simultaneous access to several HDF files from within an application, which is an important feature that the single-file interfaces do not support. It is recommended that the user explore the new interfaces

Three Levels of Interaction with the HDF File



Multifile Interfaces

SD API Stores, manages and retrieves multidimensional arrays of character or numeric data, along with their dimensions and attributes, in more than one file.

VS API Stores, manages and retrieves multivariate data stored as records in a table.

V API Creates groups of any primary HDF data structures.

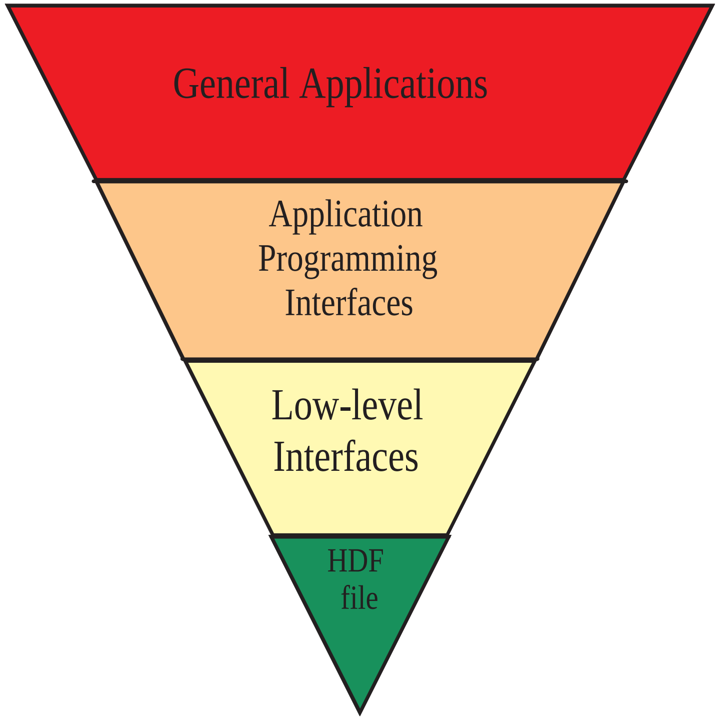
GR API Stores, manages and retrieves raster images, their dimensions and palettes in more than one file. It can also manipulate unattached palettes in more than one file.

AN API Stores, manages and retrieves text used to describe a file or any of the data structures contained in the file. This interface can operate on several files at once.

Primary HDF Platforms

Machine	Operating System
Sun Sun4	SunOS, Solaris
SGI Indy, PowerChallenge, Origin	Irix
H/P HP9000	HPUX
SGI/Cray	UNICOS
DEC Alpha	Digital Unix, OpenVMS
DEC VAX	OpenVMS
PC	Solaris86, Linux, FreeBSD
PC	Windows NT/95
Apple Power Macintosh	MacOS

HDF Software Layers



- At the highest level HDF has a collection of utilities for manipulating, viewing and analyzing HDF files. These can be implemented in a variety of tools, such as IDL .
- HDF provides a software library with high-level APIs for use in programming, and an interface to low-level structure.
- At the lowest level, HDF is a physical file format for scientific data.

Example: Goddard DAAC

The Goddard DAAC has an HDF archive that contains all necessary HDF libraries as well as documentation needed to use these libraries and routines and to install the latest version of HDF on your machine.

We have also precompiled the libraries on 10 major platforms to further ease the installation process for the user.

Goddard Data Sets

Currently we offer four complete data sets in HDF.

see: http://daac.gsfc.nasa.gov/REFERENCE_DOCS/HDF/gdaac_hdf.html

- Pathfinder AVHRR Land Data
- Coastal Zone Color Scanner
- Atmoshperic Ozone
- TOVS Pathfinder
- SeaWIFS–Level 1, Level 2, and Level 3 data products (future)

HDF Packages

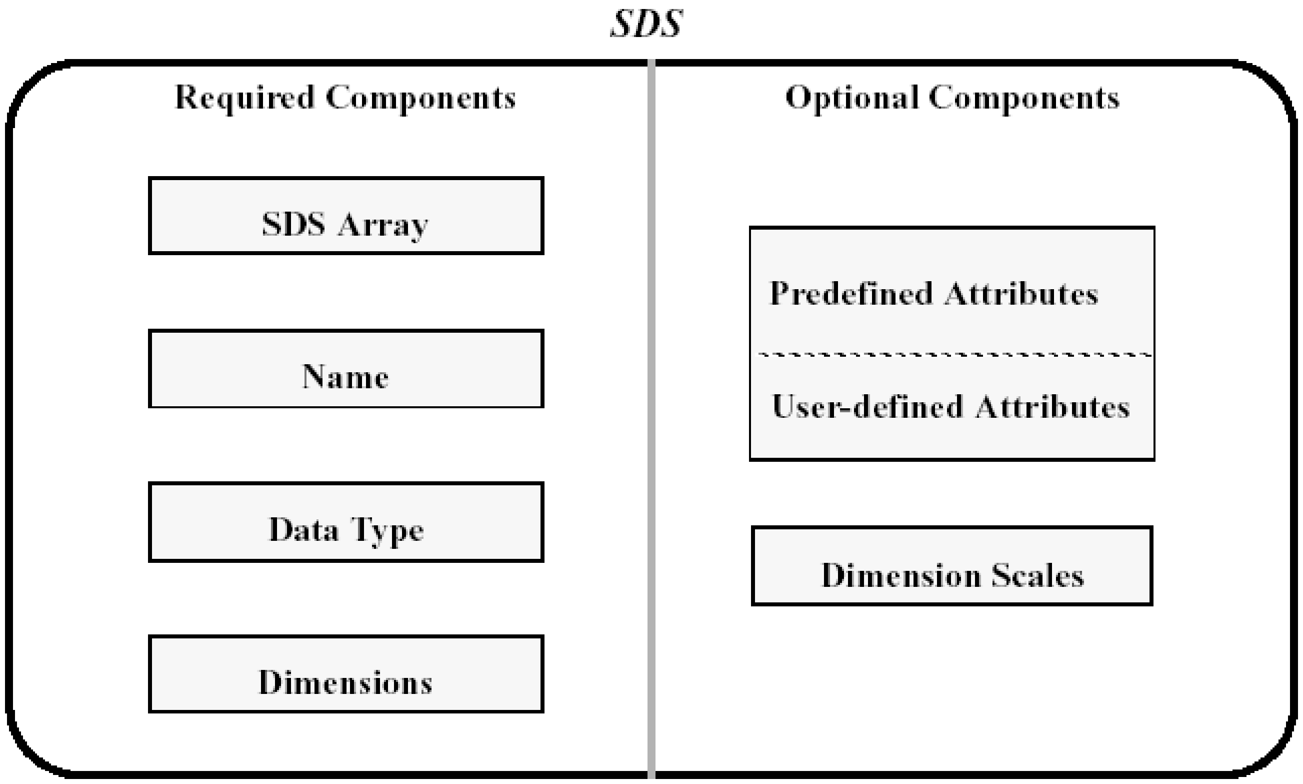
Collage NCSA has released Collage to work specifically with HDF files. Although the current version is somewhat dated, it is still recommended as a reliable tool to quickly generate and display images within an HDF file.

LinkWinds The Linked Windows Interactive Data System (LinkWinds) is a prototype visual data exploration system resulting from a NASA/JPL program of reasearch into graphical models for rapidly and interactively accessing, displaying, and analyzing large multivariate, multidisciplinary data sets. It is by far the most powerful visualization packages for HDF and can handle the global 230 MB AVHRR Land Pathfinder data files.

VISTAS 1.1 The Image Processing Applications and Development Section at NASA/JPL has released the software package VISTAS. This tool was developed to work solely with the TIROS Operation Vertical Sounder (TOVS) Pathfinder level 3 data. It acts as a ”data management” tool which helps the user to easily retrieve any parameter information within the file, generate animations, and more.

Scientific Data Sets

The scientific data set, or SDS, is a group of data structures used to store and describe multidimensional arrays of scientific data.



SDS Components

SDS array A multidimensional data structure that serves as the core structure of an SDS. This is the primary data component of the SDS model and can be compressed and/or stored in external files. SDS arrays are conceptually equivalent to variables in the netCDF data model.

- **Index** is a non-negative integer that describes the relative position of the data set in the file. A valid index ranges from 0 to the total number of data sets in the file minus 1.
- **Reference Number** is a unique positive integer assigned to the data set by the SD interface when the data set is created.
- **SDS identifier** uniquely identifies a data set within the file. The identifier is created by the SD interface access routines when a new SDS is created or an existing one is selected. The identifier is then used by other SD interface routines to access the SDS until the access to this SDS is terminated.

SDS Components (cont)

SDS Name The name of an SDS can be provided by the calling program, or is set to ”DataSet” by the HDF library at the creation of the SDS. The name is assigned only when the data set is created, and cannot be changed.

Data Type The data contained in an SDS array has a data type associated with it. The standard data types supported by the SD interface include 32- and 64-bit floating-point numbers, 8-, 16- and 32-bit signed integers, 8-, 16- and 32-bit unsigned integers, and 8-bit characters. The SD interface also allows the creation of SD data sets consisting of data elements of non-standard lengths (1 to 32 bits).

SDS Components (cont)

Dimensions specify the shape and size of an SDS array. The number of dimensions of an array is referred to as the rank of the array. Each dimension has an index and an identifier assigned to it. A dimension also has a size and may have a name associated with it.

- **Identifier** is a positive number uniquely assigned to the dimension by the library.
- **Index** is a non-negative number that describes the ordinal location of a dimension among others in a data set.
- **Names** can optionally be assigned to dimensions. If a name assigned to a dimension was previously assigned to another dimension the SD interface treats both dimensions as the same data component and any changes made to one will be reflected in the other.
- **size** of a dimension is a positive integer.One dimension of an SDS array can be assigned the predefined size SD_UNLIMITED (or 0). This dimension is referred to as an *unlimited dimension*, which can grow to any length.

SDS Optional Components

Attributes describe the nature and/or the intended usage of the file, data set, or dimension they are attached to. Attributes have a name and value which contains one or more data entries of the same data type. The name, data type, number of values, and values are specified when the attribute is created.

User-defined attributes are defined by the calling program and contain auxiliary information about a file, SDS array, or dimension.

Predefined attributes have reserved names and, in some cases, predefined data types and/or number of data entries. Predefined attributes are useful because they establish conventions that applications can depend on.

Dimension scale is a sequence of numbers placed along a dimension to demarcate intervals along it.

IDL Scientific Dataset Routines

Below is a listing of routines for the HDF Scientific Data format.

HDF_SD_ADDDATA	HDF_SD_ATTRFIND
HDF_SD_ATTRINFO	HDF_SD_ATTRSET
HDF_SD_CREATE	HDF_SD_DIMGET
HDF_SD_DIMGETID	HDF_SD_DIMSET
HDF_SD_END	HDF_SD_ENDACCESS
HDF_SD_GETDATA	HDF_SD_GETINFO
HDF_SD_IDTOREF	HDF_SD_ISCOORDVAR
HDF_SD_NAMETOINDEX	HDF_SD_REFTOINDEX
HDF_SD_SELECT	HDF_SD_SETCOMPRESS
HDF_SD_SETEXTFILE	HDF_SD_SETINFO
HDF_SD_START	HDF_SD_FILEINFO

Example

Discover the contents of a HDF file.

Beginning with the file name

- Get the file ID
- Get the number of variables and global attrbutes
    - Get the global attribute names
    - Get the variable names

Program to probe an HDF file

```
PRO probe1,fname
hdfid=HDF_SD_START(fname)
HDF_SD_FILEINFO,hdfid,nvars,ngatts

Print,fname
Print,'Number of SD global attributes=',ngatts
Print,'Number of SD variables=',nvars

Print,'***************GLOBAL ATTRIBUTES******************'
IF ngatts LE 0 THEN $
Print,'No global attributes' ELSE $
FOR attind=0L,ngatts-1L DO BEGIN
    HDF_SD_ATTRINFO,hdfid,attind,COUNT=count,DATA=data,NAME=name,TYPE=type
    Print,name
    Print,'Count=',count
    Print,'Type=',type
    Help,data
    Print,'-------------------------------------------------'
ENDFOR
```

Continued on the next page

Program to probe an HDF file (cont)

```
Print,'***************Variables**************************'
IF nvars LE 0 THEN $
    Print,'No variables' ELSE BEGIN
          varnames=strarr(nvars)
          For index=0L,nvars-1L DO BEGIN
               varid=HDF_SD_SELECT(hdfid,index)
               HDF_SD_GETINFO,varid,NAME=name
               HDF_SD_ENDACCESS,varid
               varnames[index]=name
          ENDFOR
    ENDELSE
Print,varnames
HDF_SD_END,hdfid
END
```

Probing Manassas.hdf

```
c:\rsi\idl55\gumley\data\Manassas.hdf
Number of SD global attributes=           6
Number of SD variables=           1
***************GLOBAL ATTRIBUTES******************
PRID: Profile Identification
Count=          19
Type=STRING
DATA            STRING    = 'SDTS RASTER PROFILE'
-------------------------------------------------
PDOC: Profile Document Reference
Count=          52
Type=STRING
DATA            STRING    = 'Federal Geographic Data Committee (FGDC) SDTS'...
-------------------------------------------------
PRVS: Profile Version
Count=          23
Type=STRING
DATA            STRING    = 'DRAFT VERSION JULY 1997'
-------------------------------------------------
TITL: Title
Count=          20
Type=STRING
DATA            STRING    = 'MANASSAS, VA - 24000'
-------------------------------------------------
```

Continued on the next page

Probing Manassas.hdf

```
DAID: Data Identification
Count=          40
Type=STRING
DATA            STRING    = 'LAT:: 38.75 LONG:: -77.375 SCALE:: 24000'
-------------------------------------------------
DCDT: Dataset Creation Date
Count=           8
Type=STRING
DATA            STRING    = '19980812'
-------------------------------------------------
***************Variables**************************
CEL0: ELEVATION
```

Probing EarthProbe5_31_99.hdf

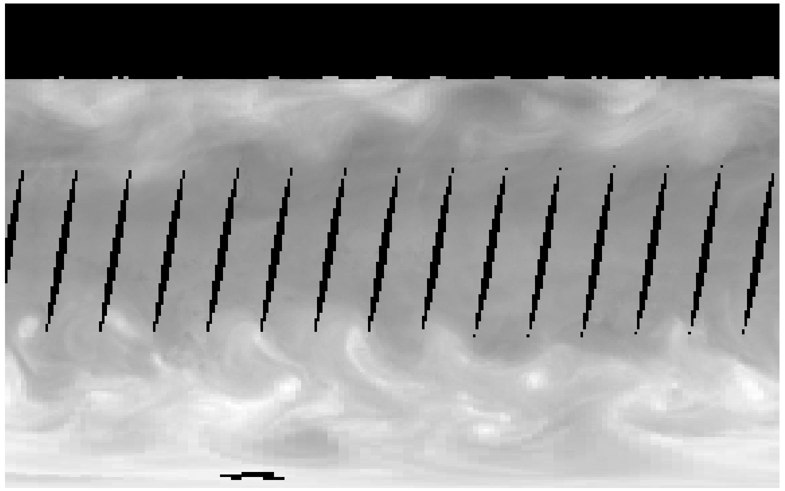
```
C:\RSI\IDL55\gumley\data\EarthProbe5_31_99.hdf
Number of SD global attributes=           0
Number of SD variables=           4
***************GLOBAL ATTRIBUTES******************
No global attributes
***************Variables**************************
TOTAL_OZONE LATITUDE LONGITUDE REFLECTIVITY
```

Get Data from EarthProbe5_31_99.hdf

```
IDL> hdfid=HDF\_SD\_START('C:\RSI\IDL55\gumley\data\EarthProbe5_31_99.hdf')
IDL> index=HDF\_SD\_NAMETOINDEX(hdfid,'TOTAL\_OZONE')
IDL> PRINT,index
           0
IDL> varid=HDF\_SD\_SELECT(hdfid,index)
IDL> HDF\_SD\_GETINFO,varid,NDIMS=ndims,DIMS=dims,TYPE=type

IDL> print,ndims,dims,type
           2         288          180
INT

IDL> hdf\_sd\_getdata,varid,ozone
IDL> help,ozone
OZONE           INT       = Array[288, 180]
IDL> window,/free,xsize=288,ysize=180
IDL> tvscl,ozone
```



Example: Data Set Construction

```
file=filepath('ctscan.dat',subdir='examples/data')
openr,lun,file,/get_lun
data=bytarr(256,256)
readu,lun,data
free_lun,lun
hdfid=HDF_SD_START(‘ctscan.hdf',/create)
varid=HDF_SD_CREATE(hdfid,‘image',[256,256],/byte)
dimid=HDF_SD_DIMGETID(varid,0)
HDF_SD_DIMSET,dimid,NAME='xdim'
dimid=HDF_SD_DIMGETID(varid,1)
HDF_SD_DIMSET,dimid,NAME='ydim'
HDF_SD_ADDDATA,varid,data
HDF_SD_ATTRSET,varid,‘long_name','CT scan from IDL examples'
HDF_SD_ENDACCESS,varid
HDF_SD_END,hdfid

probe1,'ctscan.hdf'
;ctscan.hdf
;Number of SD global attributes=           0
;Number of SD variables=          1
;**************GLOBAL ATTRIBUTES*****************
;No global attributes
;**************Variables*************************
;image
```

Example: Add to the Data Set

```
hdfid=HDF_SD_START(‘ctscan.hdf’,/RDWR)
varid=HDF_SD_CREATE(hdfid,’frequency’,[256,256],/float)
dimid=HDF_SD_DIMGETID(varid,0)
HDF_SD_DIMSET,dimid,NAME=’xdim’
dimid=HDF_SD_DIMGETID(varid,1)
HDF_SD_DIMSET,dimid,NAME=’ydim’
HDF_SD_ADDDATA,varid,dist(256)
HDF_SD_ENDACCESS,varid
HDF_SD_END,hdfid
```

```
probe1,’ctscan.hdf’
;ctscan.hdf
;Number of SD global attributes=          0
;Number of SD variables=          2
;***************GLOBAL ATTRIBUTES******************
;No global attributes
;***************Variables**************************
;image frequency
```

Example: Read from the Data Set

```
IDL> hdfid=HDF_SD_START('ctscan.hdf')
IDL> varid=HDF_SD_SELECT(hdfid,0)
IDL> HDF_SD_GETINFO,varid,DIMS=dims,NDIMS=ndims,TYPE=type
IDL> Print,dims,ndims
         256         256
           2
IDL> print,type
BYTE
IDL> HDF_SD_GETDATA,varid,x
IDL> help,x
X               BYTE      = Array[256, 256]
IDL> window,/free,xsize=256,ysize=256
IDL> tv,x
IDL> tvscl,x
IDL> pwd
C:\RSI\IDL55\gumley\demos
IDL> write_eps,'ctscan.eps',x
IDL> HDF_SD_ENDACCESS,varid
IDL> HDF_SD_END,hdfid
```