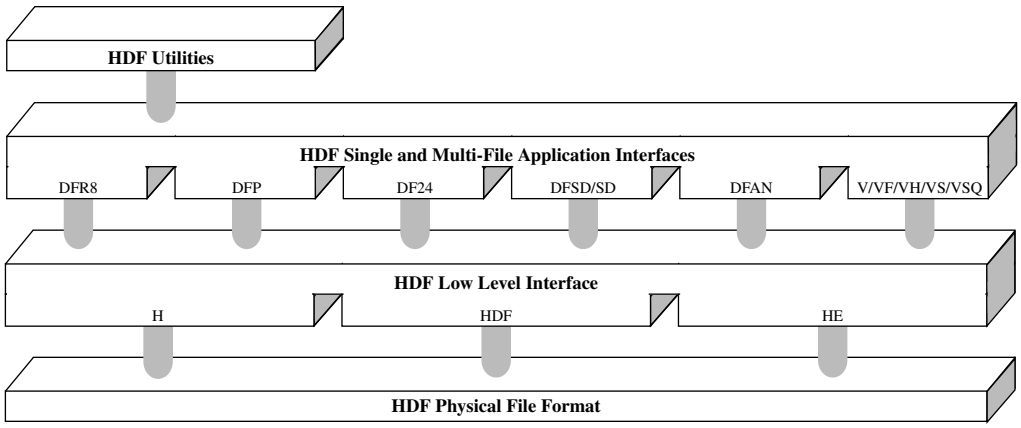
Chapter 1

Introduction to the HDF Function Calls and Utilities

1.0 Overview of the HDF Interfaces

The HDF library consist of three interface layers built upon a physical file format (See Figure 1.0). The first layer, or the ***low level interface***, is reserved for software developers because it provides support for low level details like file I/O, error handling, and memory management. The second layer, containing the ***single*** and ***multi-file application interfaces***, consists of a set of interfaces designed to simplify the process of storing and accessing data. The single file interfaces are limited to operating on one file at a time whereas the multi-file interfaces are not. Finally, the highest HDF layer includes a collection of command line ***utilities*** that operate on HDF files or the data objects they contain.

FIGURE 1.0 **The HDF interfaces and their respective layers.**



1.1 The Low Level Interface

The first interface layer, or the ***low level interface***, is the layer of HDF reserved for software developers. Through its H, HDF, and HE routines, the low level interface provides routines for error handling, file I/O, memory management, and physical storage. For a more detailed discussion of the low level interface than that provided here, consult the *HDF Specification and Developer's Guide*.

1.1.1 The H Interface

The low level ***H interface*** provides a collection of routines for managing HDF files. It is called the H interface because all the routines begin with the letter H. The H routines are divided into the following six categories:

- **File access routines** are used to open and close HDF files.
- **Reading and writing routines** access data objects and allow part or all of a data element to be written or read from an HDF file.
- **Special data object routines** create "linked blocks" and "external elements."
- **General file maintenance routines** truncate data elements to a specified length, and synchronize data on disk with data in memory**.**
- **Data descriptor routines** manipulate the data descriptor block without affecting the data objects stored in the file.
- **Query routines** locate specific data objects in a file, or return information regarding the contents of an HDF file.

Prior to HDFv3.2, all low level routines began with the prefix DF. As of HDFv3.3, the ***DF interface*** is no longer recommended for use. It is only supported to maintain backward compatibility with programs and files created under earlier version of the HDF library.

The H function calls are more explicitly identified in Table 1.1.1 and on their respective pages in Chapter 2. There are no FORTRAN routines for most of the H interface because this interface is intended primarily for the manipulation of existing C data structures.

TABLE 1.1.1 Functions Available in the H Interface

Purpose	Function Name		Description
	C	FORTRAN	
Accessing HDF files	`Hopen`	`hopen`	Provides an access path to an HDF file
	`Hclose`	`hclose`	Closes the access path to the file
Reading and writing data elements	`Hstartread`	None	Locates an object by tag and ref number and returns its descriptor
	`Hnextread`	None	Locates and positions a read access marker
	`Hstartwrite`	None	Initializes the write operation
	`Hendaccess`	None	Disposes of descriptor for tag/ref
	`Hseek`	None	Seeks to a position in a data element
	`Hread`	None	Reads part of a data element.
	`Hwrite`	None	Writes next data segment to data element.
	`Hgetelement`	None	Obtains the data referred to by the tag and ref number passed to it
	`Hputelement`	None	Adds or replaces an element in a file
Creating special data elements	`HLcreate`	None	Creates or transforms a linked block data element
	`HXcreate`	None	Creates or transforms an external file data element
General file maintenance	`Htrunc`	None	Truncate a data element to a length
	`Hsync`	None	Synchronize on-disk HDF file with image in memory

Purpose	Function Name		Description
	C	FORTRAN	
Manipulating data descriptors	Hlength	None	Returns the length of a data element in bytes
	Hoffset	None	Gets the offset (in bytes) of a data element in the file
	Hdupdd	None	Generate new references to already referenced data
	Hdeldd	None	Delete a tag and reference number from the list of data descriptors
	Hnewref	None	Returns a reference number that is guaranteed to be unique in the file
Querying HDF files	Hfind	None	Locate the next object of a search in an HDF file
	Hinquire	None	Returns access information of a data object
	Happendable	None	Allows appending to the end of a data element.
	Hishdf	None	Determine if a file is an HDF file
	Hnumber	hnumber	Find the number of occurrences of tag and reference number in file
	Hgetlibversion	None	Get version information for current HDF library
	Hgetfileversion	None	Get version information for an HDF file
	Hfidinquire	None	Inquire about a file ID

1.1.2 The HDF Interface

As HDF begins expanding its interfaces to include multi-file support for each data model, it is anticipated that the some H routines will evolve into an ***HDF interface***. These routines will use the prefix HDF. In HDFv3.3, there are only two such routines, **`HDFopen`** and **`HDFclose`**. **`HDFopen`** and **`HDFclose`** currently operate as macros for **`Hopen`** and **`Hclose`** respectively.

1.1.3 The HE Interface

The HDF library incorporates an error stack via the ***HE interface***. In addition to other functions, the HE routines add information onto the error stack, print information from the stack, and clear the stack. The HE function calls are more explicitly identified in Table 1.1.3 and on their respective pages in the reference section of this manual.

TABLE 1.1.3 Functions Available in the HE Interface

Purpose	Function Name		Description
	C	FORTRAN	
Error reporting	HEclear	None	Clears the error stack
	HEprint	heprnt	Prints the errors on the error stack to a specified file
	HEpush	None	Pushes a new error onto the stack
	HEreport	None	Adds text to the description of the most recent error
	HEstring	None	Returns the error message associated with a error code
	HEvalue	None	Returns the nth most recent error

1.2 Single File Application Interfaces

The HDF ***single file application interfaces*** include several independent modules each designed to simplify the process of storing and accessing a specific type of data. Single file interfaces are so called because access is limited to one file at a time. These interfaces support the 8-bit raster image(DFR8), 24-bit raster image (DF24), palette (DFP), scientific data (DFSD), and annotation (DFAN) models. All single file interfaces are built upon the H routines; unless otherwise specified, all the low level details can be ignored.

1.2.1 8-bit Raster Image Sets: The DFR8 Interface

The HDF ***8-bit raster interface*** provides a collection of routines for managing 8-bit raster image sets. Any 8-bit raster image accompanied by its dimension record is recognized as an ***8-bit raster image set***. Raster image sets may also include a palette.

Every function in the 8-bit raster interface begins with the prefix DFR8. The equivalent FORTRAN functions use the prefix d8. The DFR8 routines are divided into three groups:

- **Write routines** are used to add image sets to a file, create raster image sets containing palettes, and assign specific tag and reference numbers to individual raster image sets.
- **Read routines** are used to determine the dimensions of an image, read an image set, and select a specific image set.
- **General interface routines** are used to determine the reference number of a raster image set, return the number of raster image sets in a file, and return to the beginning of a file to access multiple raster image sets in sequential order.

The DFR8 function calls are more explicitly identified in Table 1.2.1 and on their respective pages in the reference section of this manual. To learn more about using the ***8-bit raster data model*** or the DFR8 interface consult *Getting Started with HDF* or the *HDF User's Guide*.

TABLE 1.2.1 Functions Available in the DFR8 Interface

Purpose	Function Name		Description
	Long Name (C/Fortran)	Short Name (Fortran)	
Writing 8-bit raster images	DFR8addimage	d8aimg	Appends another 8-bit image to a file
	DFR8putimage	d8pimg	Writes an 8-bit image to a file
	DFR8setpalette	d8spal	Sets palette for multiple 8-bit images
	DFR8writeref	d8iwref	Stores image using specified reference number
	DFR8setcompress	d8scomp	Sets the compression for the next 8-bit write operation
Reading 8-bit raster images	DFR8getdims	d8gdims	Retrieves dimensions for 8-bit image
	DFR8getimage	d8gimg	Retrieves an 8-bit image and its palette
	DFR8readref	d8irref	Locates an image with a specified reference number
General interface	DFR8lastref	d8lref	Returns reference number of last element accessed
	DFR8nimages	d8inims	Returns number of images in a file
	DFR8restart	d8first	Ignores info about the last access and restarts from beginning

1.2.2 Palettes: The DFP Interface

The HDF ***palette interface*** provides a collection of routines for managing ***palette data***. This interface is most often used for working with multiple palettes stored in a single file or palettes not specifically assigned to a raster image.

Every function in the palette interface begins with the prefix DFP. The equivalent FORTRAN functions use the prefix dp. The DFP routines are divided into three categories:

- **Write routines** are used to add palettes to HDF files, overwrite existing palettes, and assign specific tag and reference numbers to individual palettes.
- **Read routines** are used to read palettes from a file in sequential order or as specified by tag and reference number.
- **General interface routines** are used to determine the reference number of the last palette written to file, return the number of palettes in a file, and return to the beginning of a file to access multiple palettes in sequential order.

The DFP routines are more explicitly defined in Table 1.2.2 and on their respective pages in the reference section of this manual. To learn more about the ***palette model*** or the DFP interface consult *Getting Started with HDF* or the *HDF User's Guide*.

TABLE 1.2.2 Functions Available in the DFP Interface

Purpose	Function Name		Description
	Long Name (C/Fortran)	Short Name (Fortran)	
Writing palettes	`DFPaddpal`	`dpapal`	Appends a palette to a file
	`DFPputpal`	`dpppal`	Writes a new palette to a file
	`DFPwriteref`	`dpwref`	Sets the reference number for writing the next palette
Reading palettes	`DFPgetpal`	`dpgpal`	Retrieves the next palette in a file
	`DFPreadref`	`dprref`	Sets reference number for retrieving the next palette
General interface	`DFPlastref`	`dplref`	Returns the value of the last reference number read or written
	`DFPnpals`	`dpnpals`	Returns the number of palettes in a file
	`DFPrestart`	`dprest`	Ignores info about the last access and restarts from beginning

1.2.3 24-bit Raster Image Sets: The DF24 Interface

The HDF***24-bit raster interface*** provides a collection of routines for managing 24-bit raster image sets. Any 24-bit raster image array accompanied by its dimension record is recognized as a ***24-bit raster image set***.

Every function in the 24-bit raster interface begins with the prefix DF24. The equivalent FORTRAN functions use the prefix d2. DF24 routines are divided into three categories:

- **Write routines** are used to add 24-bit raster image sets to a file, define the dimensions and interlacing, and assign tag and reference numbers to 24-bit raster image sets.
- **Read routines** are used to determine the dimensions and interlacing for 24-bit images, and to read images both sequentially and via their the tag and reference numbers.

- **General interface routines** report the reference number of the last raster image set written, return the number of raster sets in a file, and restart at the beginning of a file.

The DF24 function calls are more explicitly defined in Table 1.2.3 and on their respective pages in the reference section of this manual. To learn more about the ***24-bit raster model*** or the DF24 interface, consult *Getting Started with HDF* or the *HDF User's Guide*.

TABLE 1.2.3 Functions Available in the DF24 Interface

Purpose	Function Name		Description
	Long Name (C/Fortran)	Short Name (Fortran)	
Writing 24-bit raster images	DF24addimage	d2iaimg	Appends another 24-bit image to a file
	DF24putimage	d2pimg	Writes an 24-bit image into a file
	DF24setcompress	d2scomp	Sets the compression for the next 24-bit write operation
	DF24setdims	d2sdims	Sets the dimensions of image
	DF24setil	d2setil	Sets the interlacing for the image
	DF24writeref	d2wref	Stores an image using a specified reference number
Reading 24-bit raster images	DF24getdims	d2igdims	Retrieves dimensions for 24-bit image
	DF24getimage	d2igimg	Retrieves a 24-bit image
	DF24readref	d2rref	Locates an image with a specified reference number
	DF24reqil	d2reqil	Request interlace when returning image
General interface	DF24lastref	d2lref	Returns the reference number of last element accessed
	DF24restart	d2first	Ignores info about the last access and restarts from beginning

1.2.4 Scientific Data Sets: The Single File DFSD Interface

There are two HDF interfaces that support multi-dimensional arrays: the single file DFSD interface described here, which permits access to only one file at a time, and the newer SD interface, which permits simultaneous access to more than one file. The SD interface is covered more thoroughly in the multi-file section of this manual.

The single file ***scientific data set interface*** provides a collection of routines for reading and writing arrays of arbitrary rank and number type. Any array accompanied by a record of its rank and number type qualifies as a ***scientific data set***. Scientific data sets may also include predefined ***attribute records***. Attribute records describes one particular facet of the environment from which the scientific data or its dimensions were taken.

Every function in the single file interface begins with the prefix DFSD. The equivalent FORTRAN functions use the prefix ds. DFSD routines are divided into five categories:

- **Read and write routines** create and read scientific data sets by defining their dimensions, rank, number type, fill value, data range, and calibration information.
- **Slab routines** read and write subarrays of scientific data to existing data sets.
- **Data set attribute routines** are used to describe the format, units, labels and coordinate systems assigned to a data set. All attributes are optional.
- **Dimension attribute routines** are used to read and write the format, units, labels, and scales assigned to the dimensions in a data set.

- **General interface routines** are used to determine the reference number of a data set, return the number of data sets in a file, and support hyperslab operations.

The DFSD function calls are more explicitly defined in Table 1.2.4 and on their respective pages in the reference section of this manual. For more information on using the ***scientific data model*** or interface, consult the *Getting Started with HDF* or the *HDF User's Guide*.

TABLE 1.2.4 **Functions Available in the DFSD Interface**

Purpose	Function Name		Description
	Long Name (C/Fortran)	Short Name (Fortran)	
Reading and writing scientific data sets	DFSDadddata	dsadata	Appends a data set to the file
	DFSDgetcal	dsgcal	Retrieves the calibration information for the data set
	DFSDgetdata	dsgdata	Retrieves the next data set in the file
	DFSDgetdims	dsgdims	Returns the number and dimensions for the next data set
	DFSDgetNT	dsgnt	Determines the number type for the data in the data set
	DFSDputdata	dspdata	Overwrites data in a file with new data
	DFSDsetcal	dsscal	Sets the calibration information for the data set
	DFSDsetdims	dssdims	Sets the rank and dimension for the succeeding data set
	DFSDsetNT	dssnt	Sets the number type for the data set
Reading and writing slabs of scientific data	DFSDendslab	dseslab	Terminates a read or write slab operation
	DFSDgetfillvalue	dsgfill	Retrieves the fill value used to complete the data set
	DFSDreadslab	dsrslab	Reads a sub-array of data from a data set
	DFSDsetfillvalue	dssfill	Sets the fill value to use when completing a data set
	DFSDstartslab	dssslab	Begins a read or write slab operation
	DFSDwriteslab	dswslab	Writes a sub-array of data to a data set
Working with data set attributes	DFSDgetdatalen	dsgdaln	Gets the length of the attributes assigned to the data
	DFSDgetdatastrs	dsgdast	Returns label, unit and format for data
	DFSDgetrange	dsgrang	Gets the range of values for the data set
	DFSDsetdatastrs	dssdast	Sets label, unit and format for data
	DFSDsetlengths	dsslens	Sets the length for the data set and dimension attributes
	DFSDsetrange	dssrang	Sets the range of values for the data set
Working with dimension attributes	DFSDgetdimlen	dsgdiln	Gets the length of the attributes assigned to the dimension
	DFSDgetdimscale	dsgdisc	Returns the scale for a dimension
	DFSDgetdimstrs	dsgdist	Returns the label, unit, and format for a dimension
	DFSDsetdimscale	dssdisc	Sets the scale for a dimension
	DFSDsetdimstrs	dssdist	Sets label, unit and format for the dimension
General interface	DFSDclear	dsclear	Clears all possible set values
	DFSDlastref	dslref	Returns the reference number of last data set accessed
	DFSDndatasets	dsnum	Returns the number of data sets in a file
	DFSDpre32sdg	dsp32sd	Determines if the data set was created before HDFv3.2
	DFSDreadref	dsrref	Locates a data set with a specified reference number
	DFSDrestart	dsfirst	Sets the access command to the first data set in the file
	DFSDlastref	dslref	Returns the reference number of last data set accessed
	DFSDwriteref	dswref	Assigns a specific reference number to the next data set written

1.2.5 Annotations: The DFAN Interface

The ***annotation interface*** provides a collection of routines for reading and writing text strings assigned to HDF data objects or files. Annotations consist of labels and descriptions. A ***label*** is a sequence of characters terminated by '\0'. A ***description*** is any sequence of characters which may contain one or more '\0' characters.

Every function in the annotation interface begins with the prefix DFAN. The equivalent FORTRAN functions use the prefix da. In the DFAN interface, file labels are abbreviated as ***fid***, and file descriptions are abbreviated as ***fds***. Labels and descriptions assigned to data objects are abbreviated ***lab*** and ***desc*** respectively. The DFAN routines are divided into the following three categories:

- **Write routines** are used to add file labels, file descriptions, object labels, and object descriptions to a file.
- **Read routines** are used to determine the length of an annotation, and to read annotations from a file.
- **General interface routines** are used to list the labels for a given tag, and to return the reference number of the last annotation accessed.

The DFAN function calls are more explicitly defined in Table 1.2.5 and on their respective pages in the reference section of this manual. For more information on using the ***annotation model*** or DFAN interface, consult *Getting Started with HDF* or the *HDF User's Guide*.

TABLE 1.2.5 Function Calls Available in the DFAN Interface

Purpose	Function Name		Description
	Long Name (C/Fortran)	Short Name (Fortran)	
Writing annotations	DFANputlabel	daplab	Stores an object label
	DFANputdesc	dapdesc	Stores an object description
	DFANaddfid	daafid	Adds a file label to a file
	DFANaddfds	daafds	Adds a file description to a file
Reading annotations	DFANgetlablen	dagllen	Returns length of an object label
	DFANgetlabel	daglab	Returns label object
	DFANgetdesclen	dagdlen	Returns length of object description
	DFANgetdesc	dagdesc	Returns an object description
	DFANgetfidlen	dagfidl	Returns length of file label
	DFANgetfid	dagfid	Returns file label
	DFANgetfdslen	dagfdsl	Returns length of file description
	DFANgetfds	dagfds	Returns file description
General interface	DFANlablist	dallist	Returns a list of labels for a tag type
	DFANlastref	dalref	Returns reference of last annotation accessed

1.3 Multi-File Application Interfaces

The HDF ***multi-file interfaces*** include several modules designed to allow operations on more than one file and more than one data object at the same time. The multi-file interfaces provided in HDFv3.3 are the SD, VS, VSQ, VF, V, and VH interfaces. The SD interface is the multiple file version of the scientific data set interface. It also includes expanded support for interoperability with the netCDF library. The VS, VSQ, and VF interfaces support the Vdata model and have always provided multiple file access. Similarly, the V and VH interfaces also provide multiple file access for the Vgroup data model.

Like the single file interfaces, the multi-file interfaces are built upon the low level H routines. Unlike single file operations however, operations performed via a multi-file interface are not implicitly preceded by **`Hopen`** and followed by **`Hclose`**. Instead, each series of operations on a file must be preceded by an explicit call to open and close the file. Once the file is opened, it remains open until an explicit call is made to close it. Although this process is a bit less convenient for operations on individual files, it allows for operations on more than one file at a time.

1.3.1 Scientific Data Sets: The SD Interface

The ***scientific data set interface*** provides a collection of routines for reading and writing arrays of arbitrary dimension and number type. Multi-dimensional arrays accompanied by a record of their dimension and number type are called ***scientific data sets***. Under the multi-file interface, scientific data sets may include predefined or user defined ***attribute records***. Each attribute record is optional and describes a particular facet of the environment from which the scientific data was taken.

Every function in the multi-file scientific data interface begins with the prefix SD. The equivalent FORTRAN functions use the prefix sf. The SD routines are divided into the following four categories:

- **Access routines** initialize and close the SD interface.
- **Create, read, and write routines** create, read, and write scientific data sets by defining or reading their dimensions, rank, number type, fill value, data range, and calibration information.
- **Dimension attribute routines** describe the attributes of data set dimensions including the name, format, unit, label, and scales. All attributes are optional.
- **General attribute routines** manage information about ***local attributes*** (attributes assigned to a data set) and ***global attributes*** (attributes assigned to the file). Predefined local attributes include the coordinate system, format, labels, max/min values, scales, and units. Also included are routines for user defined local and global attributes.

The SD function calls are more explicitly defined in Table 1.3.1 on page 10 and on their respective pages in the reference section of this manual. For more information about using the multi-file ***scientific data model*** or the SD interface, consult the *HDF User's Guide*.

TABLE 1.3.1 Function Calls Available in the SD Interface

Purpose	Function Name			Description
	Long Name (C/Fortran)	Short Name (Fortran)	DFSD Equivalent	
Access the SD interface	SDend	sfend	None	Close the file and clean up memory
	SDstart	sfstart	None	Initializes the SD interface
Reading and writing data sets	SDcreate	sfcreate	None	Create a new data set
	SDendaccess	sfendacc	None	Dispose of a data set ID
	SDfileinfo	sffinfo	None	Return info about the file contents
	SDgetinfo	sfginfo	None	Return information about a data set
	SDidtoref	sfid2ref	None	Returns a reference number for a named data set.
	SDiscoordvar	sfiscvar	None	Distinguishes between a data set and a coordinate variable
	SDnametoindex	sfn2index	None	Return an index for a named data set.
	SDreaddata	sfrdata	DFSDgetdata	Read a slab of data from a data set
	SDselect	sfselect	None	Gets the sds_id for a specific data set
	SDwritedata	sfwdata	DFSDadddata	Write a slab of data for a data set
Working with dimensions	SDdiminfo	sfdinfo	None	Get information about a dimension
	SDgetdimid	sfdimid	None	Get a pointer to a dimension
	SDgetdimscale	sfsdscale	DFSDgetdimscale	Retrieve scale values for a dimension
	SDsetdimname	sfsdmname	None	Associate a name with a dimension
	SDsetdimscale	sfgdscale	DFSDsetdimscale	Store the values of this dimension
Working with attributes	SDattrinfo	sfgainfo	None	Get information about an attribute
	SDfindattr	sffattr	None	Find the index for named attribute
	SDgetcal	sfgcal	DFSDgetcal	Get the calibration information
	SDgetdatastrs	sfgdtstr	DFSDgetdatastrs	Return the “data strings” for a data set
	SDgetdimstrs	sfsdmstr	DFSDgetdimstrs	Get the attribute strings for a dimension
	SDgetfillvalue	sfgfill	DFSDgetfill	Read the fill value if it exists
	SDgetrange	sfgrange	DFSDgetrange	Get the range of values for the data set
	SDreadattr	sfrattr	None	Read the values of an attribute
	SDsetattr	sfsattr	None	Define a new attribute for a variable
	SDsetcal	sfscal	DFSDsetcal	Set the calibration information
	SDsetdatastrs	sfsdtstr	DFSDsetdatastrs	Set the attribute strings for a data set
	SDsetdimstrs	sfgdmstr	DFSDsetdimstrs	Set the attribute strings for a dimension
	SDsetexternalfile	None	None	Store data in an external file
	SDsetfillvalue	sfsfill	DFSDsetfill	Set the fill value for this data set
	SDsetrange	sfsrange	DFSDsetrange	Set the max and min values

1.3.2 Scientific Data Sets: The NetCDF Interface

The SD interface is designed to be as interoperable as possible with netCDF, an interface developed by the Unidata Program Center. Consequently, the SD interface can read files written by the ***netCDF interface***, and the netCDF interface (as implemented in HDF) can read both netCDF files and HDF files that contain scientific data sets.

Further information regarding the netCDF/HDF interface can be found in Table 1.3.2 and in the *HDF User's Guide*. Additional information on the netCDF interface can be found in the *NetCDF User's Guide* available anonymous FTP from `unidata.ucar.edu`.

TABLE 1.3.2

Function Calls Available in the NC Interface

Purpose	Function Name		SD Equivalent	Description
	Long Name (C/Fortran)	Short Name (Fortran)		
Operations	`nccreate`	`NCCRE`	`SDstart`	Create a file
	`ncopen`	`NCOPN`	`SDstart`	Open a file
	`ncredef`	`NCREDF`	Not Applicable	Put open file into define mode
	`ncendef`	`NCENDF`	Not Applicable	Leave define mode
	`ncclose`	`NCCLOS`	`SDend`	Close an open file
	`ncinquire`	`NCINQ`	`SDfileinfo`	Inquire about an open file
	`ncsync`	`NCSNC`	Not Applicable	Synchronize a file to disk
	`ncabort`	`NCABOR`	Not Applicable	Back out of recent definitions
	`ncsetfill`	`NCSFIL`	Not Implemented	Set fill mode for writes
Dimensions	`ncdimdef`	`NCDDEF`	`SDsetdimname`	Create a dimension
	`ncdimid`	`NCDID`	`SDgetdimid`	Get a dimension ID from its name
	`ncdiminq`	`NCDINQ`	`SDdiminfo`	Inquire about a dimension
	`ncdimrename`	`NCDREN`	Not Implemented	Rename a dimension
Variables	`ncvardef`	`NCVDEF`	`SDcreate`	Create a variable
	`ncvarid`	`NCVID`	`SDnametoindex`	Get a variable ID from its name
	`ncvarinq`	`NCVINQ`	`SDgetinfo`	Get information about a variable
	`ncvarput1`	`NCVPT1`	Not Implemented	Write a single data value
	`ncvarget1`	`NCVGT1`	Not Implemented	Read a single data value
	`ncvarput`	`NCVPT`	`SDwritedata`	Write a hyperslab of values
	`ncvarget`	`NCVGT/NCVGTC`	`SDreaddata`	Read a hyperslab of values
	`ncvarrename`	`NCVREN`	Not Implemented	Rename a variable
	`nctypelen`	`NCTLEN`	`DFKNTsize`	Get number of bytes for a data type
Attributes	`ncattput`	`NCAPT/NCAPTC`	`SDsetattr`	Create an attribute
	`ncattinq`	`NCAINQ`	`SDattrinfo`	Get information about an attribute
	`ncattcopy`	`NCACPY`	Not Implemented	Copy attribute from one file to another
	`ncattget`	`NCAGT/NCAGTC`	`SDreadattr`	Get attributes values
	`ncattname`	`NCANAM`	`SDattrinfo`	Get name of attribute from its number
	`ncattrename`	`NCAREN`	Not Implemented	Rename an attribute
	`ncattdel`	`NCADEL`	Not Implemented	Delete an attribute

1.3.3 Vdata: The VS Interface

The ***Vdata interface*** provides a collection of routines for reading and writing customized tables. Each table is comprised of a series of ***Vdata records*** whose values are stored in fixed length fields. In addition to its records, a Vdata may contain three kinds of identifying information: a ***Vdata name***, ***Vdata class***, and a series of individual ***Vdata field names***.

Functions in the Vdata interface begins with the prefix VS. The equivalent FORTRAN functions use the prefix vsf. Vdata functions are divided into four categories:

- **Access routines** begin access to a Vdata, terminate access to a Vdata, and seek to any position in a Vdata set.
- **Specify routines** define new Vdata fields, assign a name and class, and initialize the access conditions for read and write permission**.**
- **Inquiry routines** check if a Vdata set exists, return information about particular Vdata set such as its class name or field names, its size, or whether it exists as a lone entity.
- **Read/Write routines** retrieve and store Vdata in HDF files.

The VS function calls are more explicitly defined in Table 1.3.3 and on their respective pages in the reference section of this manual. The VS interface is also supplemented by the VSQ, VF, and VH interfaces described in later sections. To learn more about the ***Vdata model*** or interface consult the *HDF User's Guide.*

TABLE 1.3.3 **Function Calls Available in the VS Interface.**

Purpose	Function Name		Description
	Long Name (C/Fortran)	Short Name (Fortran)	
Accessing the VS interface	VSattach	vsfatch	Attaches to the Vdata with the specified ID
	VSdetach	vsfdtch	Detaches from the current Vdata thereby terminating access
	VSseek	vsfseek	Reads a specified record from a Vdata
Specifying a Vdata	VSfdefine	vsffdef	Defines a new field for a Vdata
	VSsetclass	vsfscls	Assigns a class name to the Vdata
	VSsetfields	vsfsfld	Establishes permission to read or write to a field
	VSsetinterlace	vsfsint	Sets the interlace type for the Vdata
	VSsetname	vsfsnam	Assigns a name to the Vdata
Inquiring about Vdata	VSgetid	vsfgid	Returns the ID for the Vdata after the Vdata with the given ID
	VSgetname	vsfgnam	Returns the name (if any) of the specified Vdata
	VSgetfields	vsfgfld	Returns the field names (if any) for the Vdata
	VSinquire	vsfinq	This routine is the general Vdata inquiry routine
	VSlone	vsflone	Lists reference numbers for Vdata which do not belong to a Vgroup
	VSfexist	vsfex	Checks whether all the specified field(s) exists
	VSgetclass	vsfgcls	Returns the class name (if any) of the Vdata
	VSsizeof	vsfsiz	Computes the byte size of the field(s) for the local machine
	VSgetinterlace	vsfgint	Returns the interlace of the Vdata
	VSfind	vsffnd	Locates a Vdata set with a given name
Reading and writing Vdata	VSread	vsfread	Reads data from a Vdata
	VSwrite	vsfwrit	Writes data to a Vdata

1.3.4 Vdata Query: The VSQ Interface

The ***Vdata Query interface*** provides a collection of routines for inquiring about existing Vdata. These routines provide information such as the number of records in a Vdata, its field names, number types, and name. All functions in the Vdata Query interface begin with the prefix VSQ. There are no equivalent FORTRAN functions.

The VSQ function calls are more explicitly defined in Table 1.3.4 and on their respective pages in the reference section of this manual. To learn more about the ***Vdata model*** or the VSQ interface consult the *HDF User's Guide.*

TABLE 1.3.4 **Functions Available in the VSQ Interface**

Purpose	Function Name (C Only)	Description
Query for Vdata information	`VSQuerycount`	Returns the number of records in the Vdata
	`VSQueryfields`	Returns the names of the fields in the Vdata
	`VSQueryinterlace`	Returns the type of interlace in the Vdata
	`VSQueryname`	Returns the name of the Vdata
	`VSQueryref`	Retrieves the reference number for the Vdata
	`VSQuerytag`	Retrieves the tag for the specified Vdata
	`VSQueryvsize`	Retrieves the local size in bytes of a Vdata record

1.3.5 Vdata Fields: The VF Interface

The ***Vdata Fields interface*** provides a collection of routines for inquiring about the fields in an existing Vdata. These routines provide information such as the field name, size, order, and number type. All functions in the Vdata Field interface begins with the prefix VF. There are no equivalent FORTRAN functions.

The VF function calls are more explicitly defined in Table 1.3.5 and on their respective pages in the reference section of this manual. To learn more about the ***Vdata model*** or the VF interface consult the *HDF User's Guide.*

TABLE 1.3.5 **Functions Available in the VF Interface**

Purpose	Function Name (C Only)	Description
Query for Vdata field information	`VFfieldesize`	Retrieve the size (as stored in an HDF file) of a specified field in a Vdata
	`VFfieldisize`	Retrieve the size (as stored in memory) of a specified field in a give Vdata
	`VFfieldname`	Retrieves the fieldname for a specified field in a given Vdata
	`VFfieldorder`	Retrieves the order of a specified field in a given Vdata
	`VFfieldtype`	Retrieves the number type for a specified field in a given Vdata
	`VFnfields`	Retrieve the total number of fields in the specified Vdata

1.3.6 Vgroups: The V Interface

The ***Vgroup interface*** provides a collection of routines for reading and writing customized data sets. Each Vgroup may contain one or more Vdata sets, Vgroups, or data objects stored via other HDF data models. In addition to its members, a Vgroup may also be given a ***Vgroup name*** and a ***Vgroup class***.

Every function in the Vgroup interface begins with the prefix V. The equivalent FORTRAN functions use the prefix vf. The Vgroup functions are divided into four categories:

- **Access routines** open and close the V interface and begin and end access to Vgroups.
- **Specify routines** assign names, classes and members to Vgroups.
- **Inquiry routines** return the name and class of a Vgroup, determine the number, ID, and format for the members of a Vgroup.

The V function calls are more explicitly defined in Table 1.3.6 and on their respective pages in the reference section of this manual. To learn more about the ***Vgroup model*** or interface consult the *HDF User's Guide.*

TABLE 1.3.6 **Function Calls Available in the V Interface**

Purpose	Function Name		Description
	Long Name (C/Fortran)	Short Name (Fortran)	
Accessing the V interface	`Vattach`	`vfatch`	Attaches to a Vgroup when given a Vgroup ID
	`Vdetach`	`vfdtch`	Detaches the Vgroup thereby terminating access
	`Vend`	`vfend`	Opens an HDF file
	`Vstart`	`vfstart`	Closes an HDF file
Specifying a Vgroup	`Vsetclass`	`vfscls`	Assigns a class name to a Vgroup
	`Vsetname`	`vfsnam`	Assigns a name to a Vgroup
	`Vaddtagref`	`vfsdtr`	Adds an HDF object into a given Vgroup
	`Vinsert`	`vfinsrt`	Adds Vdata or Vgroups into other Vgroups
	`Vgetid`	`vfgid`	Returns the Vgroup following the Vgroup with the given ID
Inquiring about Vgroups	`Vgetclass`	`vfgcls`	Returns the class name (if any) of the Vgroup
	`Vgetname`	`vfgnam`	Returns the name (if any) of the Vgroup
	`Vinquire`	`vfinq`	This is the general Vgroup inquiry routine
	`Vlone`	`vflone`	Returns the ID for every Vgroup that is not part of another Vgroup
	`Vinqtagref`	`vfinqtr`	Tests whether an HDF object is a member of a Vgroup
	`Vntagrefs`		Returns a count of the number of HDF objects in a given Vgroup
	`Vgettagrefs`	`vfgttrs`	Returns the tag and reference number for the HDF objects in a given Vgroup
	`Vgettagref`	`vfgttr`	Returns tag and reference number for an object at a given position in a Vgroup

1.3.7 High Level VData/Vgroups: The VH Interface

The high level ***Vdata/Vgroup interface*** provides a collection of routines for creating simple Vdatas and Vgroups with a single function call. All functions in the High Level interface begins with the prefix ***VH***. There are no equivalent FORTRAN functions.

The VH function calls are more explicitly defined in Table 1.3.7 and on their respective pages in the reference section of this manual. To learn more about the Vdata or Vgroup model and the VH interface consult the *HDF User's Guide.*

TABLE 1.3.7 **Functions Available in the VH Interface**

Purpose	Function Name (C Only)	Description
Creating a Vdata or Vgroup	`VHmakegroup`	Groups a collection of tags and reference numbers into a Vgroup
	`VHstoredata`	Creates Vdatas containing single field records with one component per field
	`VHstoredatam`	Creates Vdatas containing single field records with multiple components per field

1.4 HDF Command Line Utilities

All HDF ***utilities*** are small application programs executed through a command line interface. In general, HDF utilities make it possible for you to perform common operations without requiring you to write your own programs.

Each HDF utility supports a number of command line options. These options are exercised in a similar manner to UNIX commands, i.e. they are specified by a letter preceded by a dash. For example, the **hdfls** utility will list the contents of an HDF file in ascending order if a `-o` is included on the command line. By default, **hdfls** lists information in descending numerical order.

There are currently 16 different HDF utilities, which fall into the following categories:

- **Inquiry utilities** are primarily used to examine the contents of an HDF file.
- **Conversion utilities** include a variety of programs for converting between different HDF formats and raw data formats.
- **Compression utilities** are useful for compressing HDF files using standard compression algorithms and compacting techniques.
- **Vdata/Vgroup utilities** are useful for inquiring about files that contain Vdata and Vgroups.

The HDF utilities are more explicitly defined in Table 1.4 on page 16 and on their respective pages in the reference section of this manual. To learn more about the utilities consult *Getting Started with HDF* or the *HDF User's Guide.*

TABLE 1.4 **Utilities included in the HDF distribution.**

Purpose		Utility	Definition
Inquiry		`hdfls`	Displays the tags and reference numbers for the data objects contained in a file.
		`hdfed`	Invokes and HDF editor which allows you to browse through HDF files and manipulate the data objects.
		`hdfrseq`	Displays sequences of remote 8-bit images directly to the screen.
Conversion	**Raw to HDF**	`r8tohdf`	Converts raw 8-bit images to HDF raster image sets and writes them to a file.
		`paltohdf`	Converts raw palettes to HDF format and writes them to file.
		`fp2hdf`	Converts 32-bit floating-point data to scientific data or 8-bit raster image sets.
	HDF to Raw	`hdftor8`	Converts HDF raster image sets to raw format and separates them into image files and palette files.
		`hdftopal`	Converts palettes in HDF files to raw format and writes them to separate files.
	HDF to HDF	`fp2hdf`	Converts 32-bit floating-point data to scientific data or 8-bit raster image sets.
		`hdf24to8`	Converts 24-bit HDF images to 8-bit raster image sets.
		`hdf24to8`	Converts 8-bit raster images sets into 24-bit raster image sets.
		`ristosds`	Converts several 8-bit raster images to one 3D scientific data set.
	HDF to JFIF	`hdf2jpeg`	Copies JPEG compressed images from an HDF file into separate JFIF files
	JFIF to HDF	`jpeg2hdf`	Copies a JPEG compressed image from a JFIF file into an HDF file.
Compression		`hdfpack`	Condenses HDF files by removing the empty spaces left by deleted objects.
		`hdfcomp`	Compresses 8-bit raster image in an HDF file using run-length encoding or IMCOMP compression.
Vdata/Vgroups		`vcompat`	Converts HDFv1.0 Vdata or Vgroups to HDFv2.0 or later
		`vmake`	Creates Vgroups by linking together Vdata, Vgroups, or both
		`vshow`	Dumps Vdata and Vgroup information contained in an HDF file.