* Introduction to the HDF APIs

# Overview of the HDF Interfaces

The HDF library structure consists of two interface layers and one application layer built upon a physical file format. (See Figure 1a) The first layer, or the low-level interface, is generally reserved for software developers because it provides support for low-level details such as 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 operate on one file at a time, whereas the multi-file interfaces can operate on several files simultaneously. The highest HDF layer includes various NCSA and commercial applications and a collection of command-line utilities that operate on HDF files or the data objects they contain.

* **Three Levels of Interaction with the HDF File**



## Low-Level Interface

This is the layer of HDF reserved for software developers and provides routines for error handling, file I/O, memory management, and physical storage. These routines are prefaced with ’H’. For a more detailed discussion of the low-level interface, consult the HDF Specifications and Developer’s Guide from the HDF WWW home page at http://www.hdfgroup.org.

The low-level interface provides a collection of routines that are prefaced with either ’H’, ’HE’, or ’HX’. The H routines are for managing HDF files. The HE routines provide error handlings. The HX routines are for managing HDF external files.

Prior to HDF version 3.2, all low-level routines began with the prefix ’DF’. As of HDF version 3.3, the DF interface was no longer recommended for use. It is only supported to maintain backward compatibility with programs and files created under earlier versions of the HDF library.

## Multi-file Application Interfaces

The HDF multi-file interfaces are designed to allow operations on more than one file and more than one data object at the same time. The multi-file interfaces provided are AN, GR, SD, VS, VSQ, VF, V, and VH. The AN interface is the multi-file version of the DFAN annotation interface. The GR interface is the multi-file version of the 8- and 24-bit raster image interfaces. The SD interface is the multi-file version of the scientific data set interface. The VS, VSQ, and VF interfaces support the vdata model. The V and VH interfaces provide support 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, 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. This process allows operations on more than one file at a time.

### Scientific Data Sets: SD Interface

The scientific data set interface provides a collection of routines for reading and writing arrays of data. Multidimensional 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 can be used to note or describe about the data being stored in scientific data sets.

The SD interface is designed to be as compatible as possible with netCDF, an interface developed by the Unidata Program Center and used to store and manipulate scientific data sets. Consequently, the SD interface can read files written by the netCDF interface, and the netCDF interface version 2.3.2 (as implemented in HDF) can read both netCDF files and HDF files that contain scientific data sets.

Further information regarding the netCDF interface routines and their equivalents in the HDF netCDF interface can be found in the HDF User's Guide, Section 3.14, "netCDF." Additional information on the netCDF interface can be found in the netCDF User's Guide available at http://www.unidata.ucar.edu/software/netcdf/docs/.

The names of the routines in the multi-file scientific data set interface are prefaced by ’SD’. The equivalent FORTRAN-77 routine names are prefaced by ’sf’.

### Annotations: AN Interface

The purpose of the AN multi-file annotation interface is to permit concurrent operations on a set of annotations that exist in more than one file. Annotations consist of labels and descriptions.

The C routine names of the multi-file annotation interface are prefaced by the string ’AN’ and the FORTRAN-77 routine names are prefaced by ’af’.

### General Raster Images: GR Interface

The routines in the GR interface provide multi-file operations on general raster image data sets.

The C routine names in the general raster interface have the prefix ’GR’ and the equivalent FORTRAN-77 routine names are prefaced by ’mg’.

### Vdata: The VS Interface

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

Routines in the VS interface are prefaced by ’VS’. The equivalent FORTRAN-77 routine names are prefaced by ’vsf’.

### Vdata Query: VSQ Interface

The VSQ 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 routines in the VSQ interface are prefaced by ’VSQ’. The equivalent FORTRAN-77 routine names are prefaced by ’vsq’.

### Vdata Fields: VF Interface

The VF 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 routines in the VF interface are prefaced by ’VF’. There are no equivalent FORTRAN-77 functions.

### Vgroups: V Interface

The vgroup interface provides a collection of routines for grouping and manipulating HDF data objects in the file. Each vgroup may contain one or more vdatas, vgroups, or other HDF data objects. In addition to its members, a vgroup may also be given a name and a class.

Every routine name in the vgroup interface are prefaced by ’V’. The equivalent FORTRAN-77 routine names are prefaced by ’vf’.

### Vdata/Vgroups: VH Interface

The high-level VH interface provides a collection of routines for creating simple vdatas and vgroups with a single function call. All routines in this interface are prefaced by ’VH’. The equivalent FORTRAN-77 routine names are prefaced by ’vh’.

### Vgroup Inquiry: VQ Interface

The high-level VQ interface provides one routine that returns tag information from a specified vgroup, and one routine that returns reference number information from a specified vgroup. All C routine names in this interface are prefaced by ’VQ’. The equivalent Fortran-77 routine names are prefaced by ’vq’.

## Single-File Application Interfaces

The HDF single-file application interfaces include several independent modules each is designed to simplify the process of storing and accessing a specific type of data. 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. Note that, as of version 4.2.6, these single-file interfaces were documented as deprecated interfaces, except DFP, the single-file pallete interface.

### 24-bit Raster Image Sets: DF24 Interface

The HDF 24-bit raster interface provides a collection of routines for managing 24-bit raster image sets. A 24-bit raster image set is comprised of a 24-bit raster image array and its accompanied dimension record. Raster image sets may also include a palette.

The names of the routines in the 24-bit raster interface are prefaced by ’DF24’. The equivalent FORTRAN-77 routine names are prefaced by ’d2’.

### 8-bit Raster Image Sets: DFR8 Interface

The HDF 8-bit raster interface provides a collection of routines for managing 8-bit raster image sets. An 8-bit raster image set is comprised of an 8-bit raster image array and its accompanied dimension record. Raster image sets may also include a palette.

Every function in the 8-bit raster interface begins with the prefix ’DFR8’. The equivalent FORTRAN-77 functions use the prefix ’d8’.

### Palettes: 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.

The names of the routines in the palette interface are prefaced by ’DFP’. The equivalent FORTRAN-77 routine names are prefaced by ’dp’.

### Scientific Data Sets: DFSD Interface

There are two HDF interfaces that support multidimensional arrays: the single-file DFSD interface described here, which permits access to only one file at a time, and the newer multi-file SD interface, which permits simultaneous access to more than one file. The existence of the single-file scientific data set interface is simply to support backward compatibility for previously created files and applications. It is recommended that the multi-file scientific data set interface is to be used where possible.

The single-file scientific data set interface provides a collection of routines for reading and writing arrays of data. A scientific data set is comprised of a scientific data array and its accompanied rank, name and number type. Scientific data sets may also include predefined attribute records.

The names of the routines in the single-file scientific data set interface are prefaced by ’DFSD’. The equivalent FORTRAN-77 routine names are prefaced by ’ds’.

### Annotations: DFAN Interface

The single-file 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.

The names of the routines in the single-file annotation interface are prefaced by ’DFAN’. The equivalent FORTRAN-77 routine names are prefaced by ’da’.

## FORTRAN-77 and C Language Issues

In order to make the FORTRAN-77 and C versions of each routine as similar as possible, some compromises have been made in the process of simplifying the interface for both programming languages.

### FORTRAN-77-to-C Translation

Nearly all of the HDF library code is written in C. The Fortran HDF API routines translate all parameter number types to C data types, then call the C routine that performs the main function. For example, d8aimg is the FORTRAN-77 equivalent for DFR8addimage. Calls to either routine execute the same C code that adds an 8-bit raster image to an HDF file - see the following figure.

* **Use of a Function Call Converter to Route FORTRAN-77 HDF Calls to the C Library**



### Case Sensitivity

FORTRAN-77 identifiers generally are not case sensitive, whereas C identifiers are. Although all of the FORTRAN-77 routines shown in this manual are written in lower case, FORTRAN-77 programs can generally call them using either upper- or lower-case letters without loss of meaning.

### Name Length

Because some FORTRAN-77 compilers only interpret identifier names with seven or fewer characters, the first seven characters of the FORTRAN-77 HDF routine names are unique.

### Header Files

The inclusion of header files is not generally permitted by FORTRAN-77 compilers. However, it is sometimes available as an option. On UNIX systems, for example, the macro processors m4 and cpp let the compiler include and preprocess header files. If this capability is not available, the user may have to copy the declarations, definitions, and values needed from the files dffunc.inc and hdf.inc into the user application. If the capability is available, the files can be included in the Fortran code. The files reside in the include/ subdirectory of the directory where the HDF library is installed on the user’s system.

### Number Type Specifications

When mixing machines, compilers, and languages, it is difficult to maintain consistent data type definitions. For instance, on some machines an integer is a 32-bit quantity and on others, a 16-bit quantity. In addition, the differences between FORTRAN-77 and C lead to difficulties in describing the types found in the argument lists of HDF routines. To maintain portability, the HDF library expects assigned names for all number types used in HDF routines. (See TABLE 1A)

* **Number Type Definitions**

|  |  |  |
| --- | --- | --- |
| Definition Name | Definition Value | Description |
|
| DFNT\_CHAR8 | 4 | 8-bit character type |
| DFNT\_CHAR | 4 | Same as DFNT\_CHAR8 |
| DFNT\_UCHAR8 | 3 | 8-bit unsigned character type |
| DFNT\_UCHAR | 3 | Same as DFNT\_UCHAR8 |
| DFNT\_INT8 | 20 | 8-bit integer type |
| DFNT\_UINT8 | 21 | 8-bit unsigned integer type |
| DFNT\_INT16 | 22 | 16-bit integer type |
| DFNT\_UINT16 | 23 | 16-bit unsigned integer type |
| DFNT\_INT32 | 24 | 32-bit integer type |
| DFNT\_UINT32 | 25 | 32-bit unsigned integer type |
| DFNT\_FLOAT32 | 5 | 32-bit floating-point type |
| DFNT\_FLOAT64 | 6 | 64-bit floating-point type |
| DFNT\_NINT8 | (DFNT\_NATIVE | DFNT\_INT8) | 8-bit native integer type |
| DFNT\_NUINT8 | (DFNT\_NATIVE | DFNT\_UINT8) | 8-bit native unsigned integer type |
| DFNT\_NINT16 | (DFNT\_NATIVE | DFNT\_INT16) | 16-bit native integer type |
| DFNT\_NUINT16 | (DFNT\_NATIVE | DFNT\_UINT16) | 16-bit native unsigned integer type |
| DFNT\_NINT32 | (DFNT\_NATIVE | DFNT\_INT32) | 32-bit native integer type |
| DFNT\_NUINT32 | (DFNT\_NATIVE | DFNT\_UINT32) | 32-bit native unsigned integer type |
| DFNT\_NFLOAT32 | (DFNT\_NATIVE | DFNT\_FLOAT32) | 32-bit native floating-point type |
| DFNT\_NFLOAT64 | (DFNT\_NATIVE | DFNT\_FLOAT64) | 64-bit native floating-point type |

When using a FORTRAN-77 number type that is not supported, the general practice is to use another number type of the same size. For example, an 8-bit signed integer can be used to store an 8-bit unsigned integer variable unless the code relies on a sign-specific operation.

### String and Array Specifications

In the declarations contained in the headers of FORTRAN-77 functions, the following conventions are followed:

* character\*(\*) x means that x refers to a string of an indefinite number of characters. It is the responsibility of the calling program to allocate enough space to hold the data to be stored in the string.
* real x(\*) means that x refers to an array of reals of indefinite size and of indefinite rank. It is the responsibility of the calling program to allocate an actual array with the correct number of dimensions and dimension sizes.
* <valid numeric data type > x means that x may have one of the numeric types listed in the Description column of (See Table 1A on page 5).
* <valid data type > x means that x may have any of the types listed in the Description column of (See Table 1A on page 5).

### FORTRAN-77, ANSI C and K&R C

As much as possible, we have conformed the HDF API routines to those implementations of Fortran and C that are in most common use today, namely FORTRAN-77, ANSI C and K&R C. Due to the increasing availability of ANSI C, future versions of HDF will no longer support K&R C.

As Fortran-90 is a superset of FORTRAN-77, HDF programs should compile and run correctly when using a Fortran-90 compiler.

## Error Codes

The error codes defined in the HDF library are listed in the following table.

* HDF Error Codes

|  |  |
| --- | --- |
| Error Code | Code Definition |
|
| DFE\_NONE | No error. |
| DFE\_FNF | File not found. |
| DFE\_DENIED | Access to file denied. |
| DFE\_ALROPEN | File already open. |
| DFE\_TOOMANY | Too many AID's or files open. |
| DFE\_BADNAME | Bad file name on open. |
| DFE\_BADACC | Bad file access mode. |
| DFE\_BADOPEN | Miscellaneous open error. |
| DFE\_NOTOPEN | File can't be closed because it hasn’t been opened. |
| DFE\_CANTCLOSE | fclose error |
| DFE\_READERROR | Read error. |
| DFE\_WRITEERROR | Write error. |
| DFE\_SEEKERROR | Seek error. |
| DFE\_RDONLY | File is read only. |
| DFE\_BADSEEK | Attempt to seek past end of element. |
| DFE\_PUTELEM | Hputelement error. |
| DFE\_GETELEM | Hgetelement error. |
| DFE\_CANTLINK | Cannot initialize link information. |
| DFE\_CANTSYNC | Cannot synchronize memory with file. |
| DFE\_BADGROUP | Error from DFdiread in opening a group. |
| DFE\_GROUPSETUP | Error from DFdisetup in opening a group. |
| DFE\_PUTGROUP | Error on putting a tag/reference number pair into a group. |
| DFE\_GROUPWRITE | Error when writing group contents. |
| DFE\_DFNULL | Data file reference is a null pointer. |
| DFE\_ILLTYPE | Data file contains an illegal type: internal error. |
| DFE\_BADDDLIST | The DD list is non-existent: internal error. |
| DFE\_NOTDFFILE | The current file is not an HDF file and it is not zero length. |
| DFE\_SEEDTWICE | The DD list already seeded: internal error. |
| DFE\_NOSUCHTAG | No such tag in the file: search failed. |
| DFE\_NOFREEDD | There are no free DD's left: internal error. |
| DFE\_BADTAG | Illegal WILDCARD tag. |
| DFE\_BADREF | Illegal WILDCARD reference number. |
| DFE\_NOMATCH | No DDs (or no more DDs) that match the specified tag/reference number pair. |
| DFE\_NOTINSET | Warning: Set contained unknown tag. Ignored. |
| DFE\_BADOFFSET | Illegal offset specified. |
| DFE\_CORRUPT | File is corrupted. |
| DFE\_NOREF | No more reference numbers are available. |
| DFE\_DUPDD | The new tag/reference number pair has been allocated. |
| DFE\_CANTMOD | Old element doesn’t exist. Cannot modify. |
| DFE\_DIFFFILES | Attempt to merge objects in different files. |
| DFE\_BADAID | An invalid AID was received. |
| DFE\_OPENAID | Active AIDs still exist. |
| DFE\_CANTFLUSH | Cannot flush DD back to file. |
| DFE\_CANTUPDATE | Cannot update the DD block. |
| DFE\_CANTHASH | Cannot add a DD to the hash table. |
| DFE\_CANTDELDD | Cannot delete a DD in the file. |
| DFE\_CANTDELHASH | Cannot delete a DD from the hash table. |
| DFE\_CANTACCESS | Cannot access specified tag/reference number pair. |
| DFE\_CANTENDACCESS | Cannot end access to data element. |
| DFE\_TABLEFULL | Access table is full. |
| DFE\_NOTINTABLE | Cannot find element in table. |
| DFE\_UNSUPPORTED | Feature not currently supported. |
| DFE\_NOSPACE | malloc failed. |
| DFE\_BADCALL | Routine calls were in the wrong order. |
| DFE\_BADPTR | NULL pointer argument was specified. |
| DFE\_BADLEN | Invalid length was specified. |
| DFE\_NOTENOUGH | Not enough space for the data. |
| DFE\_NOVALS | Values were not available. |
| DFE\_ARGS | Invalid arguments passed to the routine. |
| DFE\_INTERNAL | Serious internal error. |
| DFE\_NORESET | Too late to modify this value. |
| DFE\_GENAPP | Generic application level error. |
| DFE\_UNINIT | Interface was not initialized correctly. |
| DFE\_CANTINIT | Cannot initialize the interface the operation requires. |
| DFE\_CANTSHUTDOWN | Cannot shut down the interface the operation requires. |
| DFE\_BADDIM | Negative number of dimensions, or zero dimensions, was specified. |
| DFE\_BADFP | File contained an illegal floating point number. |
| DFE\_BADDATATYPE | Unknown or unavailable number type was specified. |
| DFE\_BADMCTYPE | Unknown or unavailable machine type was specified. |
| DFE\_BADNUMTYPE | Unknown or unavailable number type was specified. |
| DFE\_BADORDER | Unknown or illegal array order was specified. |
| DFE\_RANGE | Improper range for attempted access. |
| DFE\_BADCONV | Invalid number type conversion was specified. |
| DFE\_BADTYPE | Incompatible types were specified. |
| DFE\_BADSCHEME | Unknown compression scheme was specified. |
| DFE\_BADMODEL | Invalid compression model was specified. |
| DFE\_BADCODER | Invalid compression encoder was specified. |
| DFE\_MODEL | Error in the modeling layer of the compression operation. |
| DFE\_CODER | Error in the encoding layer of the compression operation. |
| DFE\_CINIT | Error in encoding initialization. |
| DFE\_CDECODE | Error in decoding compressed data. |
| DFE\_CENCODE | Error in encoding compressed data. |
| DFE\_CTERM | Error in encoding termination. |
| DFE\_CSEEK | Error seeking in an encoded dataset. |
| DFE\_MINIT | Error in modeling initialization. |
| DFE\_COMPINFO | Invalid compression header. |
| DFE\_CANTCOMP | Cannot compress an object. |
| DFE\_CANTDECOMP | Cannot decompress an object. |
| DFE\_NOENCODER | Encoder not available. |
| DFE\_NOSZLIB | SZIP library not available. |
| DFE\_COMPVERSION | Version error from zlib  Note: when Z\_VERSION\_ERROR (-6) returned from zlib. |
| DFE\_READCOMP | Error in reading compressed data.  Note: when one of the following error codes returned from zlib:  Z\_ERRNO (-1)  Z\_STREAM\_ERROR (-2)  Z\_DATA\_ERROR (-3)  Z\_MEM\_ERROR (-4)  Z\_BUF\_ERROR (-5) |
| DFE\_NODIM | A dimension record was not associated with the image. |
| DFE\_BADRIG | Error processing a RIG. |
| DFE\_RINOTFOUND | Cannot find raster image. |
| DFE\_BADATTR | Invalid attribute. |
| DFE\_BADTABLE | The nsdg table has incorrect information. |
| DFE\_BADSDG | Error in processing an SDG. |
| DFE\_BADNDG | Error in processing an NDG. |
| DFE\_VGSIZE | Too many elements in the vgroup. |
| DFE\_VTAB | Element not in vtab[]. |
| DFE\_CANTADDELEM | Cannot add the tag/reference number pair to the vgroup. |
| DFE\_BADVGNAME | Cannot set the vgroup name. |
| DFE\_BADVGCLASS | Cannot set the vgroup class. |
| DFE\_BADFIELDS | Invalid fields string passed to vset routine. |
| DFE\_NOVS | Cannot find the vset in the file. |
| DFE\_SYMSIZE | Too many symbols in the users table. |
| DFE\_BADATTACH | Cannot write to a previously attached vdata. |
| DFE\_BADVSNAME | Cannot set the vdata name. |
| DFE\_BADVSCLASS | Cannot set the vdata class. |
| DFE\_VSWRITE | Error writing to the vdata. |
| DFE\_VSREAD | Error reading from the vdata. |
| DFE\_BADVH | Error in the vdata header. |
| DFE\_VSCANTCREATE | Cannot create the vdata. |
| DFE\_VGCANTCREATE | Cannot create the vgroup. |
| DFE\_CANTATTACH | Cannot attach to a vdata or vset. |
| DFE\_CANTDETACH | Cannot detach a vdata or vset with write access. |
| DFE\_BITREAD | A bit read error occurred. |
| DFE\_BITWRITE | A bit write error occurred. |
| DFE\_BITSEEK | A bit seek error occurred. |
| DFE\_TBBTINS | Failed to insert the element into tree. |
| DFE\_BVNEW | Failed to create a bit vector. |
| DFE\_BVSET | Failed when setting a bit in a bit vector. |
| DFE\_BVGET | Failed when getting a bit in a bit vector. |
| DFE\_BVFIND | Failed when finding a bit in a bit vector. |