DFSDadddata/dsadata

intn DFSDadddata(char \**filename*, intn *rank*, int32 *dimsizes*[], VOIDP *data*)

|  |  |  |
| --- | --- | --- |
| filename | IN: | Name of the HDF file |
| rank | IN: | Number of dimensions in the data array to be written |
| dimsizes | IN: | Array containing the size of each dimension |
| data | IN: | Array containing the data to be stored |
| Purpose | Appends a scientific dataset in its entirety to an existing HDF file if the file exists. If not, a new file is created. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | In addition to appending a multidimensional array of data to an HDF file, DFSDadddata automatically stores any information pertinent to the dataset. It will not overwrite existing data in the file. The array data can be of any valid type. However, if no number type has been set by DFSDsetNT, it is assumed that the data is of type float32. | | |
|  | Calling DFSDadddata will write the scientific dataset and all associated information. That is, when DFSDadddata is called, any information set by a DFSDset\* call is written to the file, along with the data array itself. | | |
| FORTRAN | integer function dsadata(filename, rank, dimsizes, data) | | |
|  | character\*(\*) filename | | |
|  | integer rank | | |
|  | integer dimsizes(\*), data(\*) | | |

DFSDclear/dsclear

intn DFSDclear( )

|  |  |
| --- | --- |
| Purpose | Clears all values set by DFSDset\* routines. |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. |
| Description | After a call to DFSDclear, values set by any DFSDset\* call will not be written unless they have been set again. |
| FORTRAN | integer function dsclear( ) |

DFSDendslab/dseslab

intn DFSDendslab( )

|  |  |
| --- | --- |
| Purpose | Terminates a sequence of slab calls started by DFSDstartslab by closing the file. |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. |
| FORTRAN | integer function dseslab( ) |

DFSDendslice/dseslc

intn DFSDendslice( )

|  |  |
| --- | --- |
| Purpose | Terminates the write operation after storing a slice of data in a scientific dataset. |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. |
| Description | DFSDendslice must be called after all the slices are written. It checks to ensure that the entire dataset has been written, and if it has not, returns an error code. DFSDendslice is obsolete in favor of DFSDendslab. DFSDendslab is the recommended function call to use when terminating hyperslab (previously known as data slices) operations. HDF will continue to support DFSDendslice only to maintain backward compatibility with earlier versions of the library. |
| FORTRAN | integer function dseslc( ) |

DFSDgetcal/dsgcal

int32 DFSDgetcal(float64 \**cal*, float64 \**cal\_err*, float64 \**offset*, float64 \**offset\_err*, int32 \**ntype*)

|  |  |  |
| --- | --- | --- |
| cal | OUT: | Calibration factor |
| cal\_err | OUT: | Calibration error |
| offset | OUT: | Uncalibrated offset |
| offset\_err | OUT: | Uncalibrated offset error |
| ntype | OUT: | Number type of uncalibrated data |
| Purpose | Retrieves the calibration record, if there is one, attached to a scientific dataset. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | A calibration record contains four 64-bit floating point values followed by a 32-bit integer. | | |
|  | The relationship between a value iy stored in a dataset and the actual value y is defined as: | | |
|  | y = cal \* (iy - offset) | | |
|  | The variable offset\_err contains a potential error of *offset*, and *cal\_err* contains a potential error of *cal*. Currently the calibration record is provided for information only. The SD interface performs no operations on the data based on the calibration tag. | | |
|  | As an example, suppose the values in the calibrated dataset iy[] are the following integers: | | |
|  | iy[6] = {2, 4, 5, 11, 26, 81} | | |
|  | By defining cal = 0.50 and offset = -200.0 and applying the calibration formula, the calibrated dataset iy[] returns to its original form as a floating point array: | | |
|  | y[6] = {1001.0, 1002.0, 1002.5, 1005.5, 1013.0,1040.5} | | |
| FORTRAN | integer function dsgcal(cal, cal\_err, offset, offset\_err, ntype) | | |
|  | real cal, cal\_err, offset, offset\_err | | |
|  | integer ntype | | |

DFSDgetdata/dsgdata

intn DFSDgetdata(char \**filename*, intn *rank*, int32 *dimsizes*[], VOIDP *data*)

|  |  |  |
| --- | --- | --- |
| filename | IN: | Name of the file |
| rank | IN: | Number of dimensions |
| dimsizes | IN: | Dimensions of the *data* buffer |
| data | OUT: | Buffer for the data |
| Purpose | Reads the next dataset in the file. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | If the values of *rank* or *dimsizes* aren’t known, DFSDgetdims must be called to retrieve them and then use them to determine the buffer space needed for the array data. If the data type of the data in a scientific dataset isn’t known, DFSDgetNT must be called to retrieve it. Subsequent calls to DFSDgetdata (or to DFSDgetdims and DFSDgetdata) will sequentially read scientific datasets from the file. For example, if DFSDgetdata is called three times in succession, the third call reads data from the third scientific dataset in the file. | | |
|  | If DFSDgetdims or DFSDgetdata is called and there are no more scientific datasets left in the file, an error code is returned and nothing is read. DFSDrestart can be used to override this convention. | | |
| FORTRAN | integer function dsgdata(filename, rank, dimsizes, data) | | |
|  | character\*(\*) filename | | |
|  | integer rank | | |
|  | integer dimsizes(\*), data(\*) | | |

DFSDgetdatalen/dsgdaln

intn DFSDgetdatalen(intn \**label\_len*, intn \**unit\_len*, intn \**format\_len*, intn \**coords\_len*)

|  |  |  |
| --- | --- | --- |
| label\_len | OUT: | Maximum length of the label string |
| unit\_len | OUT: | Maximum length of the unit string |
| format\_len | OUT: | Maximum length of the format string |
| coords\_len | OUT: | Maximum length of the coordinate system string |
| Purpose | Retrieves the lengths of the label, unit, format, and coordinate system strings. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | The space allocated for the label, unit, format, and coordinate system strings must be at least one byte larger than the actual length of the string to account for the null termination. | | |
| FORTRAN | integer function dsgdaln(label\_len, unit\_len, format\_len, coords\_len) | | |
|  | integer label\_len, unit\_len, format\_len, coords\_len | | |

DFSDgetdatastrs/dsgdast

intn DFSDgetdatastrs(char \**label*, char \**unit*, char \**format*, char \**coordsys*)

|  |  |  |
| --- | --- | --- |
| label | OUT: | Label describing the data |
| unit | OUT: | Unit to be used with the data |
| format | OUT: | Format to be used in displaying data |
| coordsys | OUT: | Coordinate system |
| Purpose | Retrieves information about the label, unit, and format attribute strings associated with the data. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | The parameter *coordsys* gives the coordinate system that is to be used for interpreting the dimension information. | | |
| FORTRAN | integer function dsgdast(label, unit, format, coordsys) | | |
|  | character\*(\*) label, unit, format, coordsys | | |

DFSDgetdimlen/dsgdiln

intn DFSDgetdimlen (intn *dim*, intn \**label\_len*, intn \**unit\_len*, intn \**format\_len*)

|  |  |  |
| --- | --- | --- |
| dim | IN: | Dimension the label, unit, and format refer to |
| label\_len | OUT: | Length of the label |
| unit\_len | OUT: | Length of the unit |
| format\_len | OUT: | Length of the format |
| Purpose | Retrieves the length of the label, unit, and format attribute strings associated with the specified dimension. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | The space allocated to hold the label, unit, and format strings must be at least one byte larger than the actual length of the string, to account for the null termination. | | |
| FORTRAN | integer function dsgdiln(dim, label\_len, unit\_len, format\_len) | | |
|  | integer dim, label\_len, unit\_len, format\_len | | |

DFSDgetdims/dsgdims

intn DFSDgetdims(char \**filename*, intn \**rank*, int32 *dimsizes*[], intn *maxrank*)

|  |  |  |
| --- | --- | --- |
| filename | IN: | Name of the HDF file |
| rank | OUT: | Number of dimensions |
| dimsizes | OUT: | Buffer for the returned dimensions |
| maxrank | IN: | Size of the storage buffer *dimsizes* |
| Purpose | Retrieves the number of dimensions (*rank*) of the dataset and the sizes of the dimensions (*dimsizes*) for the next scientific dataset in the file. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | The *maxrank* parameter tells DFSDgetdims the size of the array that is allocated for storing the *dimsizes* array. The value of *rank* must not exceed the value of *maxrank*. | | |
|  | The allocation of a buffer for the scientific dataset data should correspond to the values retrieved by DFSDgetdims. The first value in the array *dimsizes* should equal the first dimension of the array that is allocated to hold the dataset; the second value in *dimsizes* should equal the second dimension of the dataset, and so forth. | | |
| FORTRAN | integer function dsgdims(filename, rank, dimsizes, maxrank) | | |
|  | character\*(\*) filename | | |
|  | integer rank, maxrank | | |
|  | integer dimsizes(\*) | | |

DFSDgetdimscale/dsgdisc

intn DFSDgetdimscale(intn *dim*, int32 *size*, VOIDP *scale*)

|  |  |  |
| --- | --- | --- |
| dim | IN: | Dimension this scale corresponds to |
| size | IN: | Size of the *scale* buffer |
| scale | OUT: | Array of values defining reference points along a specified dimension |
| Purpose | Gets the scale corresponding to the specified dimension. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | The DFSD interface requires the dimension scales to be of the same data type as the corresponding data. To store dimension scales of a different data type than the corresponding data, use the multi-file SD interface. | | |
| FORTRAN | integer function dsgdisc(dim, size, scale) | | |
|  | integer dim, size | | |
|  | integer scale(\*) | | |

DFSDgetdimstrs/dsgdist

intn DFSDgetdimstrs(intn *dim*, char \**label*, char \**unit*, char \**format*)

|  |  |  |
| --- | --- | --- |
| dim | IN: | Dimension this label, unit and format refer to |
| label | OUT: | Label that describes this dimension |
| unit | OUT: | Unit to be used with this dimension |
| format | OUT: | Format to be used in displaying scale for this dimension |
| Purpose | Retrieves the label, unit, and format attribute strings corresponding to the specified dimension. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | The space allocated for the label, unit, and format string must be at least one byte larger than the length of the string to accommodate the null termination. If the length is unknown when the program is written, declare the array size as 1+maxlen\_label, maxlen\_unit, or maxlen\_format after they are set by DFSDsetlengths. The maximum default string length is 255. | | |
| FORTRAN | integer function dsgdist(dim, label, unit, format) | | |
|  | integer dim | | |
|  | character\*(\*) label, unit, format | | |

DFSDgetfillvalue/dsgfill

intn DFSDgetfillvalue(VOIDP *fill\_value*)

|  |  |  |
| --- | --- | --- |
| fill\_value | OUT: | Fill value |
| Purpose | Retrieves the fill value of a DFSD scientific dataset. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | The fill value is set by DFSDsetfillvalue and returned in the variable fill\_value. Note that DFSDgetfillvalue does not take a file name as an argument. As a result, a DFSD call to initialize the file information structures is required before calling DFSDgetfillvalue. One such call is DFSDgetdims. | | |
| FORTRAN | integer function dsgfill(fill\_value) | | |
|  | character\*(\*) fill\_value | | |

DFSDgetNT/dsgnt

intn DFSDgetNT(int32 \**ntype*)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ntype | OUT: | | Number type of data in the scientific dataset | | | |
| Purpose | Retrieves the data type of the next dataset to be read. | | | | | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | | | | | |
| Description | Note that DFSDgetNT does not take a file name as an argument. As a result, a DFSD call to initialize the file information structures is required before calling DFSDgetNT. One such call is DFSDgetdims. | | | | | | |
|  | Valid values for *ntype* are of the general form DFNT\_. The following are valid symbolic names and their number types: | | | | | | |
| 32-bit float | | DFNT\_FLOAT32 | 5 |
| 64-bit float | | DFNT\_FLOAT64 | 6 |
| 8-bit signed int | | DFNT\_INT8 | 20 |
| 8-bit unsigned int | | DFNT\_UINT8 | 21 |
| 16-bit signed int | | DFNT\_INT16 | 22 |
| 16-bit unsigned int | | DFNT\_UINT16 | 23 |
| 32-bit signed int | | DFNT\_INT32 | 24 |
| 32-bit unsigned int | | DFNT\_UINT32 | 25 |
| 8-bit character | | DFNT\_CHAR8 | 4 |
| FORTRAN | integer function dsgnt(num\_type) | | | | | | |
|  | integer num\_type | | | | | | |

DFSDgetrange/dsgrang

intn DFSDgetrange(VOIDP *max*, VOIDP *min*)

|  |  |  |
| --- | --- | --- |
| max | OUT: | Maximum value stored with the scientific dataset |
| min | OUT: | Maximum value stored with the scientific dataset |
| Purpose | Retrieves the maximum and minimum values stored with the scientific dataset. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | The *max* and *min* values are set via a call to DFSDsetrange. They are not automatically stored when a dataset is written to a file. The data type of these values is the data type of the dataset array. One implication of this is that in the C version of DFSDgetrange the arguments are pointers, rather than simple variables, whereas in the FORTRAN-77 version they are simple variables of the same type as the data array. | | |
|  | Neither DFSDgetrange nor DFSDgetdata compare the *max* and *min* values stored with the dataset to the actual values in the dataset; they merely retrieve the data. As a result, the maximum and minimum values may not always reflect the actual maximum and minimum values in the dataset. In some cases the *max* and *min* values may actually lie outside the range of values in the dataset. | | |
| FORTRAN | integer function dsgrang(max, min) | | |
|  | character\*(\*) max, min | | |

DFSDgetslice/dsgslc

intn DFSDgetslice(char \**filename*, int32 *winst*[], int32 *windims*[], VOIDP *data*, int32 *dims*[])

|  |  |  |
| --- | --- | --- |
| filename | IN: | Name of HDF file |
| winst | IN: | Array containing the coordinates for the start of the slice |
| windim | IN: | Array containing the dimensions of the slice |
| data | OUT: | Array for returning slice |
| dims | OUT: | Dimensions of array data |
| Purpose | Reads part of a scientific dataset from a file. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | DFSDgetslice accesses the dataset last accessed by DFSDgetdims. If DFSDgetdims has not been called for the named file, DFSDgetslice gets a slice from the next dataset in the file. Array *winst* specifies the coordinates of the start of the slice. Array *windims* gives the size of the slice. The number of elements in *winst* and *windims* must be equal to the rank of the dataset. For example, if the file contains a three-dimensional dataset, *winst* may contain the values {2, 4, 3}, while windims contains the values {3, 1, 4} and the dims should be at least {3, 1, 4}, the same size as the slice. This will extract a 3 x 4, two-dimensional slice, containing the elements between (2, 4, 3) and (4, 4, 6) from the original dataset. | | |
|  | The *data* array is the array into which the slice is read. It must be at least as big as the desired slice. The *dims* array is the array containing the actual dimensions of the array *data*. The user assigns values to *dims* before calling DFSDgetslice. | | |
|  | All parameters assume FORTRAN-77-style one-based arrays. | | |
|  | DFSDgetslice is obsolete in favor of DFSDreadslab. DFSDreadslab is the recommended function call to use when reading hyperslabs (previously known as data slices). HDF will continue to support DFSDgetslice only to maintain backward compatibility with HDF applications built on earlier versions of the library. | | |
| FORTRAN | integer function dsgslc(filename, winst, windims, data, dims) | | |
|  | character\*(\*) filename, data | | |
|  | integer winst(\*), windims(\*), dims(\*) | | |

DFSDlastref/dslref

intn DFSDlastref( )

|  |  |
| --- | --- |
| Purpose | Retrieves the most recent reference number used in writing or reading a scientific dataset. |
| Return value | Returns the reference number for the last accessed scientific dataset if successful and FAIL (or -1) otherwise. |
| Description | DFSDlastref returns the value of the last reference number of a scientific dataset read from or written to the file. |

|  |  |
| --- | --- |
| FORTRAN | integer function dslref( ) |

DFSDndatasets/dsnum

intn DFSDndatasets(char \**filename*)

|  |  |  |
| --- | --- | --- |
| filename | IN: | Name of the HDF file |
| Purpose | Returns the number of scientific datasets in the file. | | |
| Return value | Returns the number of datasets if successful and FAIL (or -1) otherwise. | | |
| Description | In HDF version 3.3, DFSDndatasets replaced DFSDnumber. In order to maintain backward compatibility with existing HDF applications, HDF will continue to support DFSDnumber. However, it is recommended that all new applications use DFSDndatasets instead of DFSDnumber. | | |
| FORTRAN | integer function dsnum(filename) | | |
|  | character\*(\*) filename | | |

DFSDpre32sdg/dsp32sd

intn DFSDpre32sdg(char \**filename*, uint16 *ref*, intn \**ispre32*)

|  |  |  |
| --- | --- | --- |
| filename | IN: | The name of the HDF file containing the scientific dataset |
| ref | IN: | Reference number of SDG |
| ispre32 | OUT: | Pointer to results of the pre-HDF version 3.2 inquiry |
| Purpose | Tests if the scientific dataset with the specified reference number was created by an HDF library earlier than version 3.2. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | If the scientific dataset was created with a version of HDF prior to version 3.2, *ispre32* will be set to 1, otherwise it will be set to 0. Based on this information, programmers can decide whether or not to transpose the corresponding array. | | |
| FORTRAN | integer function dsp32sd(filename, ref, ispre32) | | |
|  | character\*(\*) filename | | |
|  | integer ref, ispre32 | | |

DFSDputdata/dspdata

intn DFSDputdata(char \**filename*, intn *rank*, int32 *dimsizes*[], VOIDP *data*)

|  |  |  |
| --- | --- | --- |
| filename | IN: | Name of the HDF file |
| rank | IN: | Number of dimensions of data array to be stored |
| dimsizes | IN: | Buffer for the dimension sizes |
| data | IN: | Buffer for the data to be stored |
| Purpose | Writes a scientific data and related information to an HDF file. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | DFSDputdata will write data to an existing file by destroying the contents of the original file. Use it with caution. If a new filename is used, DFSDputdata functions exactly like DFSDadddata. | | |
| FORTRAN | integer function dspdata(filename, rank, dimsizes, data) | | |
|  | character\*(\*) filename | | |
|  | <valid numeric data type> data | | |
|  | integer rank | | |
|  | integer dimsizes(\*) | | |

DFSDputslice/dspslc

intn DFSDputslice(int32 *windims*[], VOIDP *source*, int32 *dims*[])

|  |  |  |
| --- | --- | --- |
| windims | IN: | Window dimensions specifying the size of the slice to be written |
| source | IN: | Buffer for the slice |
| dims | IN: | Dimensions of the *source* array |
| Purpose | Writes part of a scientific dataset to a file. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | DFSDputslice read a subset of an array in memory and stores it as part of the scientific dataset array last specified by DFSDsetdims. Slices must be stored contiguously. | | |
|  | Array *windims* (“window dimensions”) specifies the size of the slice to be written. The *windims* array must contain as many elements as there are dimensions in the entire scientific dataset array. The *source* argument is an array in memory containing the slice and *dims* is an array containing the dimensions of the array source. | | |
|  | Notice that *windims* and *dims* need not be the same. The *windims* argument could refer to a sub-array of *source*, in which case only a portion of *source* is written to the scientific data array. | | |
|  | All parameters assume FORTRAN-77-style one-based arrays. | | |
|  | DFSDputslice is obsolete in favor of DFSDwriteslab. DFSDwriteslab is the recommended function call to use when writing hyperslabs (previously known as data slices). HDF will continue to support DFSDputslice only to maintain backward compatibility with earlier versions of the library. | | |

DFSDreadref/dsrref

intn DFSDreadref(char \**filename*, uint16 *ref*)

|  |  |  |
| --- | --- | --- |
| filename | IN: | Name of the HDF file |
| ref | IN: | Reference number for next DFSDgetdata call |
| Purpose | Specifies the reference number for the dataset to be read during the next read operation. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | This routine is commonly used in conjunction with DFANgetlablist, which returns a list of labels for a given tag together with their reference numbers. It provides a sort of random access to scientific datasets. | | |
|  | There is no guarantee that reference numbers appear in sequence in an HDF file, so it is not generally safe to assume that a reference number is an index number of a scientific dataset. | | |
| FORTRAN | integer function dsrref(filename, ref) | | |
|  | character\*(\*) filename | | |
|  | integer ref | | |

DFSDreadslab/dsrslab

intn DFSDreadslab(char \**filename*, int32 *start*[], int32 *slab\_size*[], int32 *stride*[], VOIDP *buffer*, int32 *buffer\_size*[])

|  |  |  |
| --- | --- | --- |
| filename | IN: | Name of the HDF file |
| start | IN: | Buffer of size *rank* containing the coordinates for the start of the slab |
| slab\_size | IN: | Buffer of size *rank* containing the size of each dimension in the slab |
| stride | IN: | Subsampling (not yet implemented) |
| buffer | OUT: | \Buffer for the returned slab |
| buffer\_size | OUT: | Dimensions of the *buffer* parameter |
| Purpose | Reads a slab of data from any scientific dataset. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | DFSDreadslab will access to the scientific dataset following the current one if DFSDgetdims or DFSDgetdata are not called earlier. The *start* array indices are one-based. The rank of *start* must be the same as the number of dimensions of the specified variable. The elements of *slab\_size* must be no larger than the dimensions of the scientific dataset in order. The stride feature is not currently implemented. For now just pass the *start* array as the argument for *stride* where it will be ignored. | | |
|  | To extract a slab of lower dimension than that of the dataset, enter 1 in the *slab\_size* array for each omitted dimension. For example, to extract a two-dimensional slab from a three-dimensional dataset, specify the beginning coordinates in three dimensions and enter a 1 for the missing dimension in the *slab\_size* array. More specifically, to extract a 3 *x* 4 slab containing the elements (6, 7, 8) through (8, 7, 11) specify the beginning coordinates as {6, 7, 8} and the slab size as {3, 1, 4}. | | |
| FORTRAN | integer function dsrslab(filename, start, slab\_size, stride, buffer, buffersize) | | |
|  | character\*(\*) filename, buffer | | |
|  | integer start(\*), slab\_size(\*), | | |
|  | integer stride(\*), buffer\_size(\*) | | |

DFSDrestart/dsfirst

intn DFSDrestart( )

|  |  |
| --- | --- |
| Purpose | Causes the next read command to be read from the first scientific dataset in the file, rather than the scientific dataset following the one that was most recently read. |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. |
| FORTRAN | integer function dsfirst( ) |

DFSDsetcal/dsscal

intn DFSDsetcal(float64 *cal*, float64 *cal\_err*, float64 *offset*, float64 *offset\_err*, int32 *ntype*)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| cal | IN: | | Calibration factor | | |
| cal\_err | IN: | | Calibration error | | |
| offset | IN: | | Uncalibrated offset | | |
| offset\_err | IN: | | Uncalibrated offset error | | |
| ntype | IN: | | Number type of uncalibrated data | | |
| Purpose | Sets the calibration information associated with data | | | | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | | | | |
| Description | This routine sets the calibration record associated with a dataset. A calibration record contains four 64-bit floating point values followed by a 32-bit integer, to be interpreted as follows: | | | | | |
| cal | | calibration factor |
| cal\_err | | calibration error |
| offset | | calibrated offset |
| offset\_err | | calibrated offset error |
| ntype | | number type of uncalibrated data |
|  | The relationship between a value iy stored in a dataset and the actual value y is defined as: | | | | | |
|  | y = cal \* (iy - offset) | | | | | |
|  | The variable offset\_err contains a potential error of offset, and cal\_err contains a potential error of cal. Currently the calibration record is provided for information only. The SD interface performs no operations on the data based on the calibration tag. | | | | | |
|  | DFSDsetcal works like other DFSDset\* routines, with one exception: the calibration information is automatically cleared after a call to DFSDputdata or DFSDadddata. Hence, DFSDsetcal must be called again for each dataset that is to be written. | | | | | |
|  | As an example, suppose the values in a dataset y[] are as follows:  y[6]={1001.0, 1002.0, 1002.5, 1005.5, 1013.0, 1040.5} | | | | | |
|  | By defining cal = 0.50 and offset = -200.0 and applying the calibration formula, the calibrated dataset iy[] becomes as follows:  iy[6]={2, 4, 5, 11, 26, 81} | | | | | |
|  | The array iy[] can then be stored as integers. | | | | | |
| FORTRAN | integer function dsscal(cal, cal\_err, offset, offset\_err, ntype) | | | | | |
|  | real\*8 cal, cal\_err, offset, offset\_err | | | | | |
|  | integer ntype | | | | | |

DFSDsetdatastrs/dssdast

intn DFSDsetdatastrs(char \**label*, char \**unit*, char \**format*, char \**coordsys*)

|  |  |  |
| --- | --- | --- |
| label | IN: | Label describing the data |
| unit | IN: | Unit to be used with the data |
| format | IN: | Format to be used in displaying the data |
| coordsys | IN: | Coordinate system of the data |
| Purpose | Sets the label, unit, format, and coordinate system for the next dataset written to file. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| FORTRAN | integer function dssdast(label, unit, format, coordsys) | | |
|  | character\*(\*) label, unit, format, coordsys | | |

DFSDsetdims/dssdims

intn DFSDsetdims (intn *rank*, int32 *dimsizes*[])

|  |  |  |
| --- | --- | --- |
| rank | IN: | Number of dimensions |
| dimsizes | IN: | Dimensions of the scientific dataset |
| Purpose | Sets the rank and dimension sizes for all subsequent scientific datasets written to the file. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | This routine must be called before calling either DFSDsetdimstrs or DFSDsetdimscale. DFSDsetdims need not be called if other set routines are not called and the correct dimensions are supplied in DFSDputdata or DFSDadddata. | | |
|  | If the rank or dimension sizes change, all previous set calls are cleared, except for the number type, which is set by calling **DFSDsetNT**. | | |
| FORTRAN | integer function dssdims(rank, dimsizes) | | |
|  | integer rank | | |
|  | integer dimsizes(\*) | | |

DFSDsetdimscale/dssdisc

intn DFSDsetdimscale (intn *dim*, int32 *dimsize*, VOIDP *scale*)

|  |  |  |
| --- | --- | --- |
| dim | IN: | Dimension this scale corresponds to |
| dimsize | IN: | Size of the *scale* buffer |
| scale | IN: | Buffer for the scale values |
| Purpose | Defines the scale for a dimension. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | A scale is a one-dimensional array whose values describe reference points along one dimension of the dataset. For example, a two-dimensional dataset representing points on a map could have two scales, one representing points of latitude, and the other points of longitude. | | |
| FORTRAN | integer function dssdisc (dim, dimsize, scale) | | |
|  | integer dim | | |
|  | integer dimsize(\*), scale(\*) | | |

DFSDsetdimstrs/dssdist

intn DFSDsetdimstrs(intn *dim*, char \**label*, char \**unit*, char \**format*)

|  |  |  |
| --- | --- | --- |
| dim | IN: | Dimension this label, unit and format refer to |
| label | IN: | Label that describes this dimension |
| unit | IN: | Unit to be used with this dimension |
| format | IN: | Format to be used to display scale |
| Purpose | Sets the label, unit, and format strings corresponding to the specified dimension. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | In both FORTRAN-77 and C programs, *dim* = 1 for the first dimension, and *dim* = 2 for the second dimension. If the user is not interested in one or more strings, empty strings can be used as parameters for the DFSDsetdimstrs call. For example, DFSDsetdimstrs(1, “vertical”, “ “, “ “) will set the label for the first dimension to “vertical” and set the unit and format to empty strings. | | |
| FORTRAN | integer function dssdist(dim, label, unit, format) | | |
|  | integer dim | | |
|  | character\*(\*) label, unit, format | | |

DFSDsetfillvalue/dssfill

intn DFSDsetfillvalue(VOIDP *fill\_value*)

|  |  |  |
| --- | --- | --- |
| fill\_value | IN: | Fill value |
| Purpose | Set the value used to fill in any unwritten location in a scientific dataset. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | It is assumed that the fill value has the same number type as the dataset. Once the fill value is set for a particular SDS, it cannot be changed. | | |
|  | If DFSDsetfillvalue is called before the first call to DFSDstartslab, DFSDstartslab will set the fill value tag attribute to the value specified in the DFSDsetfillvalue call, but will not actually write out the fill value when DFSDwriteslab is called. However, if DFSDsetfillvalue is called after the first call the DFSDstartslab, the fill value tag attribute will be set by DFSDsetfillvalue and the fill value will be written to the slab during the DFSDwriteslab call. | | |
| FORTRAN | integer function dssfill(fill\_value) | | |
|  | character\*(\*) fill\_value | | |

DFSDsetlengths/dsslens

intn DFSDsetlengths(intn *label\_len*, intn *unit\_len*, intn *format\_len*, intn *coords\_len*)

|  |  |  |
| --- | --- | --- |
| label\_len | IN: | Maximum length of label strings |
| unit\_len | IN: | Maximum length of unit strings |
| format\_len | IN: | Maximum length of format strings |
| coords\_len | IN: | Maximum length of coordinate system strings |
| Purpose | Sets the maximum lengths for the strings that will hold labels, units, formats, and the name of the coordinate system. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | The lengths set by this routine are used by the routines DFSDgetdimstrs and DFSDgetdatastrs to determine the maximum lengths of strings that they get from the file. | | |
|  | Normally, DFSDsetlengths is not needed. If it is not called, default maximum lengths of 255 are used for all strings. | | |
| FORTRAN | integer function dsslens(label\_len, unit\_len, format\_len, coords\_len) | | |
|  | integer label\_len, unit\_len, format\_len, coords\_len | | |

DFSDsetNT/dssnt

intn DFSDsetNT(int32 *ntype*)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ntype | IN: | | Number type | | | |
| Purpose | Sets the number type of the data to be written in the next write operation. | | | | | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | | | | | |
| Description | DFSDsetNT must be called if a number type other than float32 is to be stored. DFSDsetNT and DFSDsetdims can be called in any order, but they should be called before any other DFSDset\* functions and before DFSDputdata or DFSDadddata. | | | | | | |
|  | The following symbolic names can be used as the value of *ntype*: | | | | | | |
| 32-bit float | | DFNT\_FLOAT32 | 5 |
| 64-bit float | | DFNT\_FLOAT64 | 6 |
| 8-bit signed int | | DFNT\_INT8 | 20 |
| 8-bit unsigned int | | DFNT\_UINT8 | 21 |
| 16-bit signed int | | DFNT\_INT16 | 22 |
| 16-bit unsigned int | | DFNT\_UINT16 | 23 |
| 32-bit signed int | | DFNT\_INT32 | 24 |
| 32-bit unsigned int | | DFNT\_UINT32 | 25 |
| 8-bit character | | DFNT\_CHAR8 | 4 |
| FORTRAN | integer function dssnt(num\_type) | | | | | | |
|  | integer num\_type | | | | | | |

DFSDsetrange/dssrang

intn DFSDsetrange(VOIDP *max*, VOIDP *min*)

|  |  |  |
| --- | --- | --- |
| max | IN: | Highest value in the range |
| min | IN: | Lowest value in the range |
| Purpose | Stores the specified maximum and minimum data values. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | It is assumed that the number type of *max* and *min* is the same as the type of the data. One implication of this is that in the C version of DFSDsetrange the arguments are pointers, rather than simple variables, whereas in the FORTRAN-77 version they are simple variables of the same type as the data array. | | |
|  | This routine does not compute the maximum and minimum values; it merely stores the values it is given. As a result, the maximum and minimum values may not always reflect the actual maximum and minimum values in the data array. | | |
|  | When the maximum and minimum values are written to a file, the HDF element that holds these values is cleared, because it is assumed that subsequent datasets will have different values for max and min. | | |
| FORTRAN | integer function dssrang(max, min) | | |
|  | character\*(\*) max, min | | |

DFSDstartslab/dssslab

intn DFSDstartslab(char \**filename*)

|  |  |  |
| --- | --- | --- |
| filename | IN: | Name of the HDF file |
| Purpose | Prepares the DFSD interface to write a slab of data to a scientific dataset. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | DFSDsetdims must be called before calling DFSDstartslab. No call which involves a file open may be made after a DFSDstartslab call until DFSDendslab is called. This routine will write out the fill values if DFSDsetfillvalue is called before this routine. | | |
| FORTRAN | integer function dssslab(filename) | | |
|  | character\*(\*) filename | | |

DFSDstartslice/dssslc

intn DFSDstartslice(char \**filename*)

|  |  |  |
| --- | --- | --- |
| filename | IN: | Name of the HDF file |
| Purpose | Prepares the interface to write a data slice to the specified file. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | Before calling DFSDstartslice, DFSDsetdims must be called to specify the dimensions of the dataset to be written to the file. DFSDstartslice always appends a new dataset to an existing file. | | |
|  | Also, DFSDstartslice must be called before DFSDputslice or DFSDendslice. | | |
|  | DFSDstartslice is obsolete in favor of DFSDstartslab. DFSDstartslab is the recommended function call to use when beginning hyperslab operations. HDF will continue to support DFSDstartslice only to maintain backward compatibility earlier versions of the library. | | |
| FORTRAN | integer function dssslc(filename) | |
|  | character\*(\*) filename | |

DFSDwriteref/dswref

intn DFSDwriteref(char \**filename*, uint16 *ref*)

|  |  |  |
| --- | --- | --- |
| filename | IN: | Name of the HDF file |
| ref | IN: | Reference number for next add or put operation |
| Purpose | Specifies the reference number, *ref*, of the dataset to be overwritten next by DFSDputdata or DFSDadddata. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | DFSDwriteref verifies the refence number’s existence before returning. If a non-existent reference number is specified, an error code will be returned. | | |
|  | As this routine alters data in a destructive manner, DFSDwriteref should be used with caution. | | |
| FORTRAN | integer function dswref(filename, ref) | | |
|  | character\*(\*) filename | | |
|  | integer ref | | |

DFSDwriteslab/dswslab

intn DFSDwriteslab(int32 *start*[], int32 *stride*[], int32 *count*[], VOIDP *data*)

|  |  |  |
| --- | --- | --- |
| start | IN: | Array containing the starting coordinates of the slab |
| stride | IN: | Array containing the dimensions for subsampling |
| count | IN: | Array containing the size of the slab |
| data | IN: | Array to hold the floating point data to be written |
| Purpose | Writes a slab of data to a scientific dataset. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | The *start* indices are relative to 1. The rank of *start* must be the same as the number of dimensions of the specified variable. The elements of *start* must be no larger than the scientific dataset's dimensions in order. The stride feature is not currently implemented. For now just pass the *start* array as the argument for the *stride* parameter, where it will be ignored. | | |
|  | The rank of *count* must be the same as the number of dimensions of the specified variable. The elements of *count* must be no larger than the scientific dataset's dimensions in order. The order in which the data will be written into the specified hyperslab is with the last dimension varying fastest. The data should be of the appropriate type for the dataset. Note that neither the compiler nor HDF software can detect if the wrong type of data is used. | | |
| FORTRAN | integer function dswslab(start, stride, count, data) | | |
|  | integer start(\*), stride(\*), count(\*) | | |
|  | character\*(\*) data | | |