vsfatch

VSattach

```
int32 VSattach(int32 file_id, int32 vdata_ref, char *access)
```

file_id	IN:	File identifier returned by **Hopen**
vdata_ref	IN:	Reference number for the Vdata to open
access	IN:	Type of access; "r" for read, "w" for write.

Purpose
Attaches to an existing Vdata or creates a new Vdata with the specified reference number and returns an identifier to that Vdata.

Return value
Returns a Vdata identifier if successful and FAIL(-1) otherwise.

Description
The attach returns an identifier to the Vdata, through which all further operations on that Vdata are carried out. See also **VSdetach()**.

Access is either "r" (read) or "w" (write).

If *access* is "r", then *vdata_ref* must be a valid reference number of an existing Vdata returned from any of the Vdata search routines (e.g. **Vgetnext** or **VSgetid**). It is an error to attach to a *vdata_ref* of -1 with "r" access.

If *access* is "w", then *vdata_ref* may be a valid reference number of an existing Vdata. An existing Vdata is generally attached with "w" access to replace part of its data, or to appended new data to it. The parameter *vdata_ref* may also be -1; this is used to create a new (empty) Vdata. The default interlace for a new Vdata is FULL_INTERLACE. (This may be changed using **VSsetinterlace**).

An existing Vdata may be multiply attached for reads, but only one attach with write access to a Vdata is allowed.

FORTRAN

```
integer function vsfatch(file_id, vdata_ref, access)
   integer*4      file_id, vdata_ref
   character*1    access
```

VSdetach

vsfdtch

VSdetach

```
VOID VSdetach(int32 vdata_id)
```

vdata_id	IN:	The Vdata access identifier returned from **VSattach**

Purpose
Detaches from the current Vdata thereby terminating further access to that Vdata.

Return value
None

Description
VSdetach updates the Vdata information in the HDF file if there are any changes. All memory used for that Vdata is freed. The *vdata_id* should not be used after that Vdata is detached.

FORTRAN

```
subroutine vsfdtch(vdata_id)
    integer*4 vdata_id
```

vsfelts

VSelts

```
int32 VSelts(int32 vdata_id)
```

vdata_id	IN:	The Vdata access identifier returned from **VSattach**

Purpose — Determines the number of elements (records) in a Vdata.

Return Value — The number of elements in a Vdata if successful and `FAIL(-1)` otherwise.

Description — This routine determines the number of elements (i.e. records) in the Vdata *vdata_id*, and returns that value to the user.

FORTRAN

```
integer function vsfelts(vdata_id)
integer vdata_id
```

VSfdefine

vsffdef

VSfdefine

```
intn VSfdefine(int32 vdata_id, char *fieldname, int32 numbertype, int32 order)
```

vdata_id	IN:	The Vdata access identifier returned from **VSattach**
fieldname	IN:	Name of field to define
numbertype	IN:	Number type for the field values
order	IN:	Order of the new field

Purpose

Defines a new field for in a Vdata.

Return value

Returns SUCCEED(0) if successful and FAIL(-1) otherwise.

Description

VSfdefine is only used to define fields in a new Vdata; it does not set the format of a Vdata. Note that defining a field using **VSfdefine** does not prepare the storage format of the Vdata. Once the fields have been defined, the routine **VSsetfields** must be used to set the format. **VSfdefine** may only be used with a new (empty) Vdata. Once there is data in a Vdata, definitions of fields in the Vdata may not be modified or deleted.

A field is defined by its name (fieldname), its type (number type) and its order (order). The name of the field is any sequence of characters. By convention, this is usually mnemonic, e.g. "PRESSURE". The type of a field specifies whether a field is float, integer, etc. Thus, *numbertype* may be one of the following:

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

The order of a field is the number of components in that field. Single variables like time or pressure have an order of 1. Compound variables have an order greater than 1. For example, the variable VELOCITY has an order of 3, since velocity has three components.

vsffdef

Example

The following two calls to **VSfdefine** define 2 fields in the newly created Vdata. The first field, named VEL, is a 3-component float-valued field, while the second field, named PRESSURE, is a single-component float-valued field.

```
Vdata = (int32) VSattach(hdf_file, -1, "w");
VSfdefine(Vdata, "VEL", DFNT_FLOAT32, 3);
VSfdefine(Vdata, "PRESSURE", DFNT_FLOAT32, 1);
```

FORTRAN

```
integer function vsffdef(vdata_id, fieldname, numbertype, order)
    integer*4      vdata_id, numbertype, order
    character*(*)  fieldname
```

VSfexist

vsfex

VSfexist

```
intn VSfexist(int32 vdata_id, char *fields)
```

vdata_id	IN:	The Vdata access identifier returned from **VSattach**
fields	IN:	Names of the fields to check for

Purpose: Checks to see if the specified fields exist in the current Vdata.

Return value: A value of `1` is returned if all field(s) exist; otherwise `FAIL(-1)` is returned

Description: **VSfexist** checks if a set of fields exist in a specified Vdata. The *fields* argument is a string of comma-separated fieldnames (e.g. "PX,PY,PZ).

FORTRAN

```
integer function vsfex(vdata_id, fields)
   integer*4      vdata_id
   character*(*)  fields
```

vsffnd

VSfind

```
int32 VSfind(int32 file_id, char *vdata_name)
```

file_id	IN:	File identifier returned by **Hopen**
vdata_name	IN:	Name of the Vdata

Purpose — Looks in the HDF file for a Vdata with a given name.

Return value — Returns the Vdata reference number if successful and `FAIL(-1)` otherwise.

Description — If there are several Vdatas with the same name, **VSfind** will only find the reference number of the first Vdata in the file with that name.

FORTRAN

```
integer function vsffnd(file_id, vdata_name)
   integer*4      file_id
   character*(*)  vdata_name
```

VSgetclass

vsfgcls

VSgetclass

```
VOID VSgetclass(int32 vdata_id, char *vdata_class)
```

vdata_id	IN:	The Vdata access identifier returned from **VSattach**
vdata_class	OUT:	Pointer to space to store the class name

Purpose — Returns the class name (if any) of the Vdata.

Return value — None.

Description — VSgetclass retrieves the class name of the specified Vdata and places it in the memory pointed to by *vdata_class*. The maximum length of the class name is defined by the macro VSNAMELENMAX.

FORTRAN

```
subroutine vsfgcls(vdata_id, vdata_class)
   integer*4      vdata_id
   character*(*)  vdata_class
```

VSgetfields

```
int32 VSgetfields(int32 vdata_id, char *fields)
```

vdata_id	IN:	The Vdata access identifier returned from **VSattach**
fields	OUT:	Storage for field names

Purpose

Retrieves the field names (if any) of all the fields in a Vdata.

Return value

Returns the number of fields in the Vdata if successful and FAIL(-1) otherwise.

Description

This routine retrieves the field names (if any) of all the fields in the specified Vdata. The field names are returned as a comma-separated string (e.g., "PX,PY,PZ").

FORTRAN

```
integer function vsfgfld(vdata_id, fields)
   integer*4      vdata_id
   character*(*)  fields
```

VSgetid

vsfgid

VSgetid

```
int32 VSgetid(int32 file_id, int32 vdata_ref)
```

file_id	IN:	File identifier returned by **Hopen**
vdata_ref	IN:	Reference number for the Vdata preceding the Vdata of interest

Purpose — Sequentially searches through an HDF file for Vdatas.

Return value — Returns the reference number for the next Vdata if successful and `FAIL(-1)` otherwise.

Description — **VSgetid** searches through the specified HDF file and returns the reference number of the next Vdata after the Vdata that has reference number *vdata_ref*. This routines is generally used to search the file sequentially for Vdatas.

To initiate a search, this routine may be called *vdata_ref* equal to -1 for. Doing so returns the reference number of the first Vdata in the file. Searching past the last Vdata in a file will return an error.

FORTRAN

```
integer function vsfgid(file_id, vdata_ref)
    integer*4 file_id, vdata_ref
```

VSgetinterlace

```
int32 VSgetinterlace(int32 vdata_id)
```

vdata_id	IN:	The Vdata access identifier returned from **VSattach**

Purpose Retrieves the interlace of a Vdata.

Return value Returns FULL_INTERLACE or NO_INTERLACE if successful and FAIL(-1) otherwise.

Description This inquiry routine returns the interlace type of the Vdata *vdata_id*.

FORTRAN

```
integer function vsfgint(vdata_id)
    integer*4 vdata_id
```

VSgetname

vsfgnam

VSgetname

```
VOID VSgetname(int32 vdata_id, char *vdata_name)
```

vdata_id	IN:	The Vdata access identifier returned from **VSattach**
vdata_name	OUT:	Storage space for Vdata name

Purpose
Retrieves the name of a Vdata.

Return value
None.

Description
VSgetname returns in *vdata_name* the name (if any) of the specified Vdata.The maximum length of the class name is defined by the macro VSNAMELENMAX. If there is no name for the Vdata, a null string "" is placed in *vdata_name*.

FORTRAN

```
subroutine vsfgnam(vdata_id, vdata_name)
   integer*4      vdata_id
   character*(*)  vdata_name
```

vsfinq

VSinquire

```
intn VSinquire(int32 vdata_id, int32 *n_records, int32 *interlace, char *fields,
          int32 *vdata_size, char *vdata_name)
```

vdata_id	IN:	The Vdata access identifier returned from **VSattach**
n_records	OUT:	The number of records in the Vdata
interlace	OUT:	Type of interlace: FULL_INTERLACE or NO_INTERLACE
fields	OUT:	A comma-separated string listing all fields in the Vdata (e.g. "PX, PY, PZ")
vdata_size	OUT:	The size in bytes of a record in the Vdata for the local machine.
vdata_name	OUT:	The name (if any) of the Vdata

Purpose: General Vdata inquiry routine.

Return value: Returns SUCCEED(0) if successful and FAIL(-1) otherwise.

Description: **VSinquire** retrieves the information for each of the variables listed above. If any of the parameters are NULL, then that data is not retrieved. See also: **VSgetfields**, **VSgetinterlace**, **VSsizeof**, and **VSgetname**.

FORTRAN:

```
integer function vsfinq(vdata_id, n_records, interlace, fields,
vdata_size, vdata_name)
   integer*4      vdata_id, n_records, interlace, vdata_size
   character*(*)  fields, vdata_name
```

VSlone

vsflone

VSlone

```
int32 VSlone(int32 file_id, int32 ref_array[], int32 maxsize)
```

file_id	IN:	File identifier returned by **Hopen**
ref_array	OUT:	Array to store a list of Vdata reference numbers
maxsize	IN:	Maximum number of Vdata reference numbers to store in the array

Purpose

Returns an array of reference numbers of all Vdatas that are not linked to any Vgroup in the HDF file.

Return value

Returns the total number of lone Vdata in a file if successful and `FAIL(-1)` otherwise.

Description

This routine is provided for applications to locate all lone Vdatas (i.e. those that are not grouped with other objects) in a HDF file.

You must provide the integer array *ref_array*, and you must specify the maximum size of the array in the argument *maxsize*. The return value from this function will be the total number of Vdatas that are not linked to any Vgroup in the file. But at most *maxsize* reference numbers will be returned in *ref_array*.

An array size of 65000 integers for *ref_array* will be usually more than adequate. The preferred method is to use dynamic memory instead; first call **VSlone** with a value of 0 for *maxsize*, and then use the returned value to allocate memory for *ref_array* to be passed to a subsequent call to **VSlone**. See also **Vlone**.

Example

The sample code illustrates the preferred method of using **VSlone**. The second call to **VSlone** returns the required reference numbers.

```
int32 file_id;     /* id of opened HDF file */
int32 maxsize, status;
int32 *ref_array;

maxsize = VSlone (file_id, NULL, 0);
ref_array = (int32*) malloc(sizeof(int32)*maxsize);
status   = VSlone (file_id, ref_array, maxsize);
```

FORTRAN

```
integer function vsflone(file_id, ref_array, maxsize)
   integer*4      file_id
   integer*4      ref_array(*)
   integer*4      maxsize
```

VSread

```
int32 VSread(int32 vdata_id, uint8 *databuf, int32 n_records,
          int32 interlace)
```

vdata_id	IN:	The Vdata access identifier returned from **VSattach**
databuf	OUT:	Buffer to store the retrieved data
n_records	IN:	Number of records to retrieve
interlace	IN:	Interlacing for the data stored in the buffer

Purpose — Retrieves data from a Vdata.

Return value — Returns the total number of records read if successful and `FAIL(-1)` otherwise.

Description — Given a Vdata *vdata_id* (with read access), and a buffer, *databuf*, this routine returns the data read from that Vdata in the buffer.

You must specify in *n_records* the number of records to be read, You must also specify the buffer interlace in *interlace*. (use `FULL_INTERLACE` (recommended) or `NO_INTERLACE`).

To read data from a Vdata, typically, the Vdata is first attached with read access, and then **VSsetfields** is called to specify which field(s) are to be read from that Vdata. Each subsequent call to **VSread** after that will return records sequentially from that Vdata.

If the call is successful, the data returned in *databuf* is laid out as specified by *interlace*, and the data fields appear in the order specified in the last call to **VSsetfields** for that Vdata. To retrieve any arbitrary record from a Vdata, use **VSseek** to specify the record position before calling **VSread**. See also **VSseek**, and **VSsetfields**.

FORTRAN

```
integer function vsfread(vdata_id, databuf, n_records,
interlace)
    integer*4       vdata_id, n_records, interlace
    character*(*)   databuf
```

VSseek

vsfseek

VSseek

```
int32 VSseek(int32 vdata_id, int32 record)
```

vdata_id	IN:	The Vdata access identifier returned from **VSattach**
record	IN:	Record number to seek to

Purpose
Provides a mechanism for random-access I/O within a Vdata.

Return value
Returns the record position (zero or a positive integer) if successful and `FAIL(-1)` otherwise.

Description
This routine moves the access pointer within the Vdata *vdata_id* to the position of the record specified by *record*. The next **VSread** or **VSwrite** will read the current record. For example, to seek to the third record in the Vdata, set *record* to `2`. The first record position is specified by specifying a record of `0`. Each seek is constrained to a record boundary within the Vdata.

FORTRAN

```
integer function vsfseek(vdata_id, record)
    integer*4 vdata_id, record
```

vsfscls

VSsetclass

```
VOID VSsetclass(int32 vdata_id, char *vdata_class)
```

vdata_id	IN:	The Vdata access identifier returned from **VSattach**
vdata_class	IN:	Name of the Vdata class

Purpose
Sets the class name of the Vdata.

Return value
None.

Description
Vdatas initially have a class name of NULL. The class name may be reset more than once. Class names, like Vdata names, can be of any character strings. They exist solely as meaningful labels to applications and are not used by the HDF library in any way. See also **VSgetclass**.

FORTRAN

```
subroutine vsfscls(vdata_id, vdata_class)
   integer*4      vdata_id
   character*(*)  vdata_class
```

VSsetfields

vsfsfld

VSsetfields

```
intn VSsetfields(int32 vdata_id, char *fields)
```

vdata_id	IN:	The Vdata access identifier returned from **VSattach**
fields	IN:	Name of Vdata fields to access

Purpose

Establishes the fields to be accessed within the Vdata by any subsequent read or write routine (**VSread** or **VSwrite**).

Return value

Returns `SUCCEED(0)` if successful and `FAIL(-1)` otherwise.

Description

Fields are specified by the string argument *fields*, a comma-separated list of fieldnames (e.g. "PX,PY,PZ"). The combined width of the fields in a Vdata must be less than 32,000 bytes. If an attempt to create a larger record is made, **VSsetfields** will return an `FAIL`.

The fieldnames allowed depends on the access of the Vdata: If access is "`r`", the fieldnames must be one of the names of the fields that existed in the Vdata when that Vdata was created. If access is "`w`", the fieldnames can be any fieldname that has been defined by calling **VSfdefine**. Users may define any fields to be used in a Vdata. Several fieldnames have already been predefined, and may be used without having to define them. See the *HDF User's Guide* for details.

VSsetfields must be called before any called to **VSread** or **VSwrite** is allowed. It sets up the data translation tables for accessing the data in a Vdata.

For reading from a Vdata, a call to **VSsetfields** sets up the fields that are to be retrieved from records in the Vdata.

For writing to a Vdata, **VSsetfields** can only be called once, to set up the fields (i.e. format) of the records in a Vdata. Once the Vdata format is set, it may not be changed.

FORTRAN

```
integer function vsfsfld(vdata_id, fields)
   integer*4      vdata_id
   character*(*)  fields
```

VSsetinterlace

```
intn VSsetinterlace(int32 vdata_id, int32 interlace)
```

vdata_id	IN:	The Vdata access identifier returned from **VSattach**
interlace	IN:	Interlace for storing data in an HDF file

Purpose
Sets the interlace type for a Vdata.

Return value
Returns SUCCEED(0) if successful and FAIL(-1) otherwise.

Description
VSsetinterlace determines the interlace for the data written to the specified Vdata. The interlace may be specified by either FULL_INTERLACE or NO_INTERLACE. If this routine is not called the default interlace type is FULL_INTERLACE. The FULL_INTERLACE option is more efficient than NO_INTERLACE although both require the same amount of disk space. This routine can only be used when creating new Vdatas with write access.

Specifying FULL_INTERLACE fills the Vdata by record, i.e. all fields in a record are filled before moving to the next record. Specifying NO_INTERLACE fills the Vdata by field, i.e. the first field in each record is filled before moving to next series of fields.

FORTRAN

```
integer function vsfsint(vdata_id, interface)
    integer*4 vdata_id, interlace
```

VSsetname

vsfsnam

VSsetname

```
VOID VSsetname(int32 vdata_id, char *vdata_name)
```

vdata_id	IN:	The Vdata access identifier returned from **VSattach**
vdata_name	IN:	Name for the Vdata

Purpose
Assigns a name to a Vdata.

Return value
None.

Description
Vdatas initially have a Vdata name of NULL. The name may be reset more than once. Vdata names, like class names, can be of any character strings. They exist solely as a meaningful label for other applications and are not used by the HDF library in any way. See also **VSgetname**.

FORTRAN

```
subroutine vsfsnam(vdata_id, vdata_name)
   integer*4      vdata_id
   character*(*)  vdata_name
```

VSsizeof

```
int32 VSsizeof(int32 vdata_id, char *fields)
```

vdata_id	IN:	The Vdata access identifier returned from **VSattach**
fields	IN:	Name(s) of the fields to check

Purpose

Computes the byte size of the given field(s) for the local machine.

Return value

Returns the local machine size in bytes for the field if successful and FAIL(-1) otherwise.

Description

Specify a single field or several fields (comma-separated) in the parameter *fields*. The field(s) should already exist for that Vdata *vdata_id*.

This routine is useful in determining how much memory needs to be allocated to read in the field(s) in question.

Example

The code segment below uses **VSsizeof** to compute the byte size of a data value of the "DENSITY" field, and uses **VSQuerycount** to get the total number of records in the Vdata. The product gives the total memory size needed to read all the "DENSITY" data values from the Vdata.

```
int32 varsize, n_records;
unsigned char *readbuffer;
int32 vdata_id;

VSQuerycount(vdata_id, &n_records);
varsize = VSsizeof(vdata_id,"DENSITY");
readbuffer = (unsigned char*) malloc(varsize * n_records);
```

FORTRAN

```
integer function vsfsiz(vdata_id, fields)
   integer*4      vdata_id
   character*(*)  fields
```

VSwrite

vsfwrit

VSwrite

```
int32 VSwrite(int32 vdata_id, unsigned char *databuf, int32 n_records,
          int32 interlace)
```

vdata_id	IN:	The Vdata access identifier returned from **VSattach**
databuf	IN:	Buffer of records to store in the Vdata
n_records	IN:	Number of records to store
interlace	IN:	Interlace of the buffer in memory

Purpose
Writes data into a Vdata.

Return value
Returns the total number of records written if successful and FAIL(-1) otherwise.

Description
Given a buffer containing data to be written and a Vdata with write access, this routine writes the data from the buffer to the Vdata. The n_records argument specifies the number of record to be written and interlace defines the order in which the Vdata fields are written. Selecting FULL_INTERLACE fills the Vdata by record, i.e. all fields in a record are filled before moving to the next record. FULL_INTERLACE is recommended for speed and efficiency. Specifying NO_INTERLACE fills the Vdata by field, i.e. the first field in each record is filled before moving to next series of fields.

Before writing data to a newly created Vdata, define the format of the Vdata via a call to **VSsetfields**. If the Vdata already exists, **VSsetfields** is unnecessary because data can only be written to a Vdata in compliance with its existing format.

The data in databuf is assumed to be organized in memory as specified by the interlace argument. It is also assumed to contain the exact number and order of data fields defined in the last call to **VSsetfields**. **VSwrite** writes the contents of databuf contiguously to the Vdata, therefore any "padding" or non-data spaces used for record or field alignment must be removed before attempting to write from databuf to the Vdata.

The interlace specifies how the data exists within the buffer, *databuf*. If the data is to be stored in a Vdata with an interlace different from that of the buffer, **VSsetinterlace** must be called prior to **VSwrite**. In most cases this is unnecessary since the default interlace type of FULL_INTERLACE is used. See also **VSsetfields**, **VSseek**, **VHstoredata**.

FORTRAN

```
integer function vsfwrit(vdata_id, databuf, n_records,
interlace)
   integer*4      vdata_id, n_records, interlace
   character*(*)  databuf
```