VSappendable/vsapp

int32 VSappendable(int32 *vdata_id*, int32 *block_size*)

vdata_id	IN:	Vdata identifier
block_size	IN:	Standard block size of appended data

Purpose Specifies that the vdata referred to by *vdata_id* can be appended to.

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

Description NOTE: This routine is obsolete.

FORTRAN

```
integer function vsapp(vdata_id, block_size)

integer vdata_id, block_size
```

VSattach/vsfatch

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:	Access mode

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` (or `-1`) otherwise.

Description **VSattach** returns an identifier to the vdata, through which all further operations on that vdata are carried out. See also **VSdetach**.

Valid values for *access* are: "`r`" for read access and "`w`" for write access.

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 an vdata with a *vdata_ref* of `1` with "`r`" access.

If *access* is "`w`", then *vdata_ref* must be the 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 file_id, vdata_ref
character*1 access
```

VSattrinfo/vsfainf

intn VSattrinfo(int32 *vdata_id*, int32 *field_index,* int32 *attr_index,* char **attr_name,* int32 **data_type,* int32 **count,* int32 **size*)

vdata_id	IN:	Vdata identifier returned from **VSattach**
field_index	IN:	Index of the target field
attr_index	IN:	Index of the target attribute
attr_name	OUT:	Name of the target attribute
data_type	OUT:	Data type of the target attribute
count	OUT:	Attribute count
size	OUT:	Size of the target attribute

Purpose Returns the name, data type, number of values, and the size of the values of the specified attribute of the specified vdata field or vdata.

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

Description The values of the *attr_name*, *data_type*, *count* and *size* parameters can be set to `NULL`, if the information returned by these parameters are not needed.

The *field_index* in **VSattrinfo** is the same as with the *field_index* parameter in **VSsetattr**. It can be set to either a integer field index, or `_HDF_ENTIRE_VDATA` to specify the vdata referred to by *vdata_id*.

FORTRAN

```
integer function vsfainf(vdata_id, field_index, attr_index,
    attr_name, data_type, count, size)

integer vdata_id, field_index, attr_index
character* (*) attr_name
integer data_type, count, size
```

VSdetach/vsfdtch

int32 VSdetach(int32 *vdata_id*)

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

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

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

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

```
integer function vsfdtch(vdata_id)

integer vdata_id
```

VSelts/vsfelts

int32 VSelts(int32 *vdata_id*)

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

Purpose Determines the number of records in the specified vdata.

Return value The number of elements in the vdata if successful and `FAIL` (or `-1`) otherwise.

FORTRAN

```
integer function vsfelts(vdata_id)

integer vdata_id
```

VSfdefine/vsffdef

intn VSfdefine(int32 *vdata_id*, char **fieldname*, int32 *data_type*, int32 *order*)

vdata_id	IN:	Vdata access identifier returned from **VSattach**
fieldname	IN:	Name of field to be defined
data_type	IN:	Data type of the field values
order	IN:	Order of the new field

Purpose Defines a new field for in a vdata.

Return value Returns `SUCCEED` (or `0`) if successful and `FAIL` (or `-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 (*data_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, *data_type* may be one of the following:

```
8-bit character DFNT_CHAR8          4
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, as velocity has three components.

Example The following two calls to **VSfdefine** define two fields in the newly-created vdata. The first field, named VEL, is a three-component float-valued field, while the second field, named PRESSURE, is a single-component float32-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, data_type,
        order)

integer vdata_id, data_type, order
character* (*) fieldname
```

VSfexist/vsfex

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

vdata_id	IN:	Vdata 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` (or `-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 vdata_id
character* (*) fields
```

VSfind/vsffnd

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

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

Purpose Queries the specified HDF file for a vdata with a given name.

Return value Returns the vdata reference number if successful and `0` otherwise.

Description If there is more than one vdata 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 file_id
character* (*) vdata_name
```

VSfindex/vsffidx

intn VSfindex(int32 *vdata_id*, char **fieldname*, int32 **findex*)

vdata_id	IN:	Vdata identifier returned from **VSattach**
fieldname	IN:	Name of the field
findex	OUT:	Returned index of the specified field

Purpose Retrieves the index of the specified field in the specified vdata.

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

Description This routine searches for field names only. It doesn't search the vdata name - use **VSinquire** or **VSgetname** to do this.

FORTRAN

```
integer function vsffidx(vdata_id, fieldname, findex)

integer vdata_id, findex
character* (*) fieldname
```

VSfindattr/vsffdat

intn VSfindattr(int32 *vdata_id*, int32 *field_index,* char **attr_name*)

vdata_id	IN:	Vdata identifier returned from **VSattach**
field_index	IN:	Index of the target field
attr_name	OUT:	Name of the target attribute

Purpose Returns the index of an attribute with the specified name.

Return value Returns the index of the target attribute if successful and `FAIL` (or `-1`) otherwise.

Description The attribute must to attached to the vdata identified by the the specified vdata identifier. If *field_index* is set to `_HDF_ENTIRE_VDATA`, the index of the attribute attached to the vdata referred to by the value of *vdata_id* is returned. When using the Fortran-77 version of this routine, specify a *field_index* value of `-1` to return the vdata attribute indices.

As with **VSsetattr**, if *field_index* is set to a zero-based integer value, the value will be used as the index of the target vdata field, and the index of the attribute with the name specifiedby the *attr_name* parameter will be returned.

FORTRAN

```
integer function vsffdat(vdata_id, field_index, attr_name)

integer vdata_id, field_index
character* (*) attr_name
```

VSfindclass

int32 VSfindclass(int32 *file_id*, const char **vdata_class*)

file_id	IN:	File identifier returned by **Hopen**
vdata_class	IN:	Class of the vdata

Purpose Returns the reference number of the first vdata coresponding to the specified vdata class.

Return value Returns the reference number of the vdata if successful and `0` or an error code otherwise.

VSfnattrs/vsffnas

int32 VSfnattrs (int32 *vdata_id*, int32 *field_attrs*)

vdata_id	IN:	Vdata identifier returned by **Vattach**
field_attrs	OUT:	Index of the target field

Purpose Returns the number of attributes attached to the specified vdata field *or* the number of attributes attached to the vdata identified by *vdata_id*.

Return value Returns the number of attributes assigned to this vdata or its fields when successful, and `FAIL` (or `-1`) otherwise.

Description This routine is different from the **VSnattrs** routine, which returns the number of attributes of the specified vdata *and* the fields contained in it.

If *field_index* is set to `_HDF_ENTIRE_VDATA`, the number of attributes attached to the vdata referred to by the value of *vdata_id* is returned. When using the Fortran-77 version of this routine, specify a *field_index* value of `-1` to return the number of vdata attributes.

As with **VSsetattr**, if *field_index* is set to a zero-based integer value, it will be used as the index of the target vdata field, and the attributes attached to that field will be returned.

FORTRAN

```
integer function vsffnas(vdata_id, field_index)

integer vdata_id, field_index
```

VSfpack/vsfcpak/vsfnpak

intn VSfpack(int32 *vdata_id*, intn *action*, char **fields_in_buf*, VOIDP *buf*, intn *buf_size*, intn *n_records*, char **fields*, VOIDP *bufptrs*[])

vdata_id	IN:	Vata identifier returned from **VSattach**
action	IN:	Action to be performed
fields_in_buf	IN:	Fields in *buf* to write to or read from the vdata
buf	IN/OUT:	Buffer for the vdata values
buf_size	IN:	Buffer size in bytes
n_records	IN:	Number of records to pack or unpack
fields	IN:	Names of the fields to be packed or unpacked. It may be a subset of the *fields_in_buf*. `NULL` stands for all fields in *buf*
bufptrs	IN/OUT:	Array of pointers to the field buffers

Purpose Packs field data into a buffer or unpacks buffered field data into vdata fields.

Return value Returns `SUCCEED` (or `0`) if successful, `FAIL` (or `-1`) otherwise..

Description The calling program should supply the number of field buffers corresponding to the number of fields to be packed or unpacked and is responsible to allocate sufficient space for each buffer. This should be at least *n_records**(total size of all fields specified in *fields_in_buf*) in length.

Valid values of *action* are: `_HDF_VSPACK` (or `0`) which pack field values into *buf*, or `_HDF_VSUNPACK` (or `1`) which unpacks vdata values into field buffers.

There are two Fortran-77 versions of **VSfpack**: **vsfnpak** and **vsfcpak**. The **vsfnpak** routine packs or unpacks a numeric field and **vsfcpak** a character type field. These routines pack one field at a time.

When **VSfpack** is called to pack field values into *buf*, *fields_in_buf* should contain all fields of the vdata. `NULL` can be used in C programs for *fields_in_buf*, a blank character ' ' in Fortran-77 programs. When **VSfpack** is called to unpack field values, *fields_in_buf* may be a subset of the vdata fields. `NULL` can be passed into **VSfpack** and the space character in **vsfnpak** and **vsfcpak** as the *fields_in_buf* parameter to specify all fields in the vdata.

If the *fields_in_buf* parameter is set to `NULL`, all fields in the vdata will be packed.

FORTRAN

```
integer function vsfnpak(vdata_id, action, fields_in_buf,
                  buf, buf_size, n_records, fields, bufptrs)

integer vdata_id, action, buf, buf_size, n_records, bufptrs
character* (*) fields_in_buf, fields


integer function vsfcpak(vdata_id, action, fields_in_buf,
                  buf, buf_size, n_records, fields, bufptrs)

integer vdata_id, action, buf, buf_size, n_records
character* (*) fields_in_buf, fields, bufptrs
```

VSgetattr/vsfgnat/vsfgcat

intn VSgetattr(int32 *vdata_id*, int32 *field_index,* int32 *attr_index,* VOIDP *values*)

vdata_id	IN:	Vdata identifier returned from **VSattach**
field_index	IN:	Index of the target field
attr_index	IN:	Index of the target attribute
values	OUT:	Buffer containing the attribute values

Purpose Returns the values of the specified attribute of the specified vdata field or vdata.

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

Description The *attr_index* parameter is the ordinal, zero-based (as with *field_index* in **VSsetattr** and this routine) index of the target attribute.

If *field_index* is set to `_HDF_ENTIRE_VDATA`, the value of the attribute attached to the vdata referred to by the value of *vdata_id* is returned. If *field_index* is set to a zero-based integer value, the value will be used as the index of the target vdata field, and the attribute located at the ordinal position specified by *attr_index* will be returned.

FORTRAN

```
integer function vsfgnat(vdata_id, field_index, attr_index,
        values)

integer vdata_id, field_index, attr_index
<valid numeric data type> values(*)


integer function vsfgcat(vdata_id, field_index, attr_index,
        values)

integer vdata_id, n_records, interlace
character* (*) values
```

VSgetclass/vsfgcls

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

vdata_id	IN:	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 Returns `SUCCEED` (or `0`) if successful and `FAIL` (or `-1`) otherwise.

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

```
integer function vsfgcls(vdata_id, vdata_class)

integer vdata_id
character* (*) vdata_class
```

VSgetfields/vsfgfld

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

vdata_id	IN:	Vdata access identifier returned from **VSattach**
fields	OUT:	Buffer for the 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` (or `-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 vdata_id
character* (*) fields
```

VSgetid/vsfgid

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` (or `-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 routine is generally used to sequentially search the file for vdatas.

To initiate a search, this routine must be called with *vdata_ref* equal to `-1`. Doing so returns the reference number of the first vdata in the file. Searching past the last vdata in a file will result in an error condition.

FORTRAN

```
integer function vsfgid(file_id, vdata_ref)

integer file_id, vdata_ref
```

VSgetinterlace/vsfgint

int32 VSgetinterlace(int32 *vdata_id*)

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

Purpose Returns the interlace type of the vdata specified by *vdata_id*.

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

FORTRAN

```
integer function vsfgint(vdata_id)

integer vdata_id
```

VSgetname/vsfgnam

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

vdata_id	IN:	Vdata access identifier returned from **VSattach**
vdata_name	OUT:	Buffer for the vdata name

Purpose Retrieves the name of the specified vdata.

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

Description **VSgetname** returns in *vdata_name* the name, if any, of the specified vdata.The maximum length of the class name is defined by `VSNAMELENMAX`. If there is no name for the vdata, a null string is placed in the *vdata_name* parameter.

FORTRAN

```
integer function vsfgnam(vdata_id, vdata_name)

integer vdata_id
character* (*) vdata_name
```

VSgetversion/vsgver

int32 VSgetversion(int32 *vdata_id*)

vdata_id	IN:	Vdata identifier

Purpose Retrieves the version number of a vdata.

Return value Returns the version number if successful and either `FAIL` (or `-1`) or an error code otherwise.

FORTRAN

```
integer vsgver(vdata_id)

integer vdata_id
```

VSinquire/vsfinq

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

vdata_id	IN:	Vdata access identifier returned from **VSattach**
n_records	OUT:	Number of records in the vdata
interlace	OUT:	Interlace type
fields	OUT:	Comma-separated string listing all fields in the vdata (e.g. "PX, PY,PZ")
vdata_size	OUT:	Size, in bytes, of a record in the vdata for the local machine
vdata_name	OUT:	Name, if any, of the vdata

Purpose General vdata inquiry routine.

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

Description **VSinquire** retrieves the information for each of the variables listed above. If any of the parameters are NULL, that data is not retrieved. Refer to the Reference Manual pages on **VSgetfields**, **VSgetinterlace**, **VSsizeof** and **VSgetname**.

Valid values for *interlace* are: FULL_INTERLACE or NO_INTERLACE.

FORTRAN

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

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

VSisattr/vsfisat

intn VSisattr(int32 *vdata_id*)

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

Purpose Determines whether the specified vdata contains attribute data.

Return value Returns `TRUE` if the vdata is an attribute, and `FALSE` otherwise.

Description As attributes are stored by the HDF library as vdatas, a means of testing whether or not a particular vdata is attribute data is needed, and provided by the **VSisattr** routine.

FORTRAN

```
integer function vsfisat(vdata_id)

integer vdata_id
```

VSlone/vsflone

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

file_id	IN:	File identifier returned by **Hopen**
ref_array	OUT:	Buffer for the returned list of vdata reference numbers
maxsize	IN:	Maximum number of vdata reference numbers to be stored in the array

Purpose Locates all lone vdatas (i.e. those that are not grouped with other objects) in a HDF file.

Return value Returns the total number of lone vdatas in the file if successful and `FAIL` (or `-1`) otherwise.

Description The integer array *ref_array* must be provided, and the maximum size of the array must be specified in the parameter *maxsize*. At most *maxsize* reference numbers will be returned in *ref_array*.

An array size of 65,000 integers for *ref_array* is generally 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**. Refer to the Reference Manual page on **Vlone**.

Example This 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 file_id, maxsize
integer ref_array(*)
```

VSnattrs/vsfnats

intn VSnattrs(int32 *vdata_id*)

vdata_id	IN:	Vdata identifier returned by **Vattach**

Purpose Returns the number of attributes of the specified vdata *and* the vdata fields contained in it.

Return value Returns the number of attributes if successful and `FAIL` (or `-1`) otherwise.

Description This routine is different from the **VSfnattrs** routine, which returns the number of attributes of a field contained in the specified vdata *or* the specified vdata.

FORTRAN

```
integer function vsfnats(vdata_id)

integer vdata_id
```

VSread/vsfrd/vsfrdc

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

vdata_id	IN:	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 mode of the data stored in the buffer

Purpose Retrieves data from the specified vdata.

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

Description The number of records to be read must be specified in *n_records* and the interlace mode of the data in the buffer must be specified in *interlace*. Use `FULL_INTERLACE` (recommended) or `NO_INTERLACE`.

The vdata is first attached with read access, then **VSsetfields** is called to specify which field(s) are to be read from that vdata. Each subsequent call to **VSread** will return successive records from that vdata.

If the call is successful, the data returned in *databuf* is formatted according to the specified interlace mode, 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**. Refer to the Reference Manual pages on **VSseek** and **VSsetfields**.

Note that there are two Fortran-77 versions of this routine; one for buffered numeric data (**vsfrd**) and the other for buffered character data (**vsfrdc**).

FORTRAN

```
integer function vsfrd(vdata_id, databuf, n_records, interlace)

integer vdata_id, n_records, interlace
<valid numeric data type> databuf(*)


integer function vsfrdc(vdata_id, databuf, n_records,
                        interlace)

integer vdata_id, n_records, interlace
character*(*) databuf
```

VSseek/vsfseek

int32 VSseek(int32 *vdata_id*, int32 *record*)

vdata_id	IN:	Vdata access identifier returned from **VSattach**
record	IN:	Target record number

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` (or `-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 call to **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* value of `0`. Each seek is constrained to a record boundary within the vdata.

FORTRAN

```
integer function vsfseek(vdata_id, record)

integer vdata_id, record
```

VSsetattr/vsfsnat/vsfscat

intn VSsetattr(int32 *vdata_id*, int32 *findex,* char **attr_name,* int32 *data_type,* int32 *count,*
VOIDP **values*)

vdata_id	IN:	Vdata identifier returned from **VSattach**
findex	IN:	Number determined by assigning each field in a record a number starting with 0
attr_name	IN:	Name of the attribute
data_type	IN:	Data type of the attribute
count	IN:	Number of attribute values
values	OUT:	Buffer containing the returned values of the attribute

Purpose Sets an attribute of a vdata or the field of a vdata.

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

Description If the field already has an attribute with the same name, the current values will be replaced with the new values if the new data type and order are the same as the current ones. Any changes in the field data type or order will result in an error condition.

An *findex* value of `_HDF_VDATA` (0xffffffff) represents the entire vdata.

FORTRAN

```
integer function vsfsnat(vdata_id, findex, attr_name,
    data_type, count, values)

integer vdata_id, findex, data_type, count, values
character* (*) attr_name


integer function vsfscat(vdata_id, findex, attr_name,
    data_type, count, values)

integer vdata_id, findex, data_type, count
character* (*) attr_name, values
```

VSsetclass/vsfscls

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

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

Purpose Sets the class name of the vdata.

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

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. Refer to the Reference Manual page on **VSgetclass**.

FORTRAN

```
integer function vsfscls(vdata_id, vdata_class)

integer vdata_id
character* (*) vdata_class
```

VSsetexternalfile/vsfsextf

intn VSsetexternalfile(int32 *vdata_id*, char **filename*, int32 *offset*)

vdata_id	IN:	Vdata identifier returned from **VSattach**
filename	IN:	Name of the external file
offset	IN:	Offset, in bytes, of the location in the external file the new data is to be written

Purpose Stores vdata information in an external file.

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

Description Only the data will be stored externally. Attributes and all metadata will remain in the primary HDF file.

IMPORTANT: The calling program must ensure that the external files are relocated along with the primary file.

Read the Reference Manual page on **SDsetexternalfile** for more information on using the external file feature.

FORTRAN

```
integer function vsfsextf(vdata_id, filename, offset)

integer vdata_id, offset
character* (*) filename
```

VSsetfields/vsfsfld

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

vdata_id	IN:	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` (or `0`) if successful and `FAIL` (or `-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 `MAX_FIELD_SIZE` 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 field-name 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. If an empty vdata is attached for a read, **VSsetfields** will return `FAIL`.

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 vdata_id
character* (*) fields
```

VSsetinterlace/vsfsint

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

vdata_id	IN:	Vdata access identifier returned from **VSattach**
interlace	IN:	Interlace mode for storing data

Purpose Sets the interlace mode of the specified vdata.

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

Description **VSsetinterlace** determines the interlace mode for the field 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 mode 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 vdata_id, interlace
```

VSsetname/vsfsnam

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

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

Purpose Assigns a name to a vdata.

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

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. Refer to the Reference Manual page on **VSgetname**.

FORTRAN

```
integer function vsfsnam(vdata_id, vdata_name)

integer vdata_id
character* (*) vdata_name
```

VSsizeof/vsfsiz

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

vdata_id	IN:	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` (or `-1`) otherwise. If one than one field is specified, **VSsizeof** will return the sum of the sizes of all of the fields.

Description Specify a single field or several fields (comma-separated) in the parameter fields. The field or fields 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 or fields 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 vdata_id
character* (*) fields
```

VSwrite/vsfwrt/vsfwrtc

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

vdata_id	IN:	Vdata access identifier returned from **VSattach**
databuf	IN:	Buffer of records to be stored in the vdata
n_records	IN:	Number of records to be stored
interlace	IN:	Interlace mode of the buffer in memory

Purpose Writes data to the specified vdata.

Return value Returns the total number of records written if successful and `FAIL` (or `-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 interlace mode of the vdata fields to bewritten to. 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 the *databuf* parameter 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. This can be done either by the calling program or by **VSfpack**. The latter is recommended.

The *interlace* parameter 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. Refer to the Reference Manual page on **VSsetfields**, **VSseek**, **VHstoredata**.

Note that there are two Fortran-77 versions of this routine; one for buffered numeric data (**vsfwrt**) and the other for buffered character data (**vsfwrtc**).

FORTRAN

```
integer function vsfwrt(vdata_id, databuf, n_records,
        interlace)

integer vdata_id, n_records, interlace
<valid numeric data type> databuf(*)


integer function vsfwrtc(vdata_id, databuf, n_records,
        interlace)

integer vdata_id, n_records, interlace
character* (*) databuf
```