VHstoredata/vhfsd/vhfscd

int32 VHstoredata(int32 *file_id*, char **fieldname*, uint8 *buf*[], int32 *n_records*, int32 *data_type*, char **vdata_name*, char **vdata_class*)

file_id	IN:	File identifier returned by **Hopen**
fieldname	IN:	Field name for the new vdata
buf	IN:	Buffer containing the records to be stored
n_records	IN:	Number of records to be stored
data_type	IN:	Type of data to be stored
vdata_name	IN:	Name of the vdata to be created
vdata_class	IN	Class of the vdata to be created

Purpose Creates and writes to a single-field vdata.

Return value Returns reference number of the newly-created vdata if successful, and `FAIL` (or `-1`) otherwise.

Description **VHstoredata** creates a single-field vdata in the file, *file_id*, and stores data from the buffer *buf* in it. Vdata name, class name and data type are specified by the parameters *vdata_name, vdata_class*, and *data_type*, respectively. Number of records in a vdata is specified by the parameter *n_records*. Field name is specified by the parameter *fieldname.*

Vstart must precede **VHstoredata**. It is not necessary, however, to call **VSattach** or **VSdetach** in conjunction with **VHstoredata**.

This routine provides a high-level method for creating single-order, single-field vdatas.

Note that there are two FORTRAN-77 versions of this routine; one for numeric data (**vhfsd**) and the other for character data (**vhfsdc**).

FORTRAN

```
integer function vhfsd(file_id, fieldname, buf, n_records,
                          data_type,
                      vdata_name, vdata_class)



integer file_id, n_records, data_type

character*(*) vdata_name, vdata_class, fieldname

<valid numeric data type> buf(*)



integer function vhfscd(file_id, fieldname, buf, n_records,
                          data_type,
                                   vdata_name, vdata_class)


               integer file_id, n_records, data_type

               character*(*) vdata_name, vdata_class, fieldname

               character*(*) buf
```

VHstoredatam/vhfsdm/vhfscdm

int32 VHstoredatam(int32 *file_id*, char **fieldname*, uint8 *buf*[], int32 *n_records*, int32 *data_type*, char **vdata_name*, char **vdata_class*, int32 *order*),

file_id	IN:	File identifier returned by **Hopen**
fieldname	IN:	Field name
buf	IN:	Buffer containing the records to be stored
n_records	IN:	Number of records to be stored
data_type	IN:	Type of data to be stored
vdata_name	IN:	Name of the vdata to be created
vdata_class	IN:	Class of the vdata to be created
order	IN:	Field order

Purpose Creates and writes to a multi-order, single-field vdata.

Return value Returns the reference number of the newly created vdata if successful, and `FAIL` (or `-1`) otherwise.

Description **VHstoredatam** creates a vdata with the name specified by the parameter *vdata_name* and a class name specified by the parameter *vdata_class* in the file identified by the parameter *file_id*. The data type of the vdata is specified by the parameter *data_type*. The vdata contains one field with the name specified by the parameter *fieldname*. The order of the field, *order*, indicates the number of vdata values stored per field. The vdata contains the number of records specified by the parameter *n_records*. The *buf* parameter should contain *n_records* records that will be stored in the vdata.

Vstart must precede **VHstoredatam**. It is not necessary, however, to call **VSattach** or **VSdetach** in conjunction with **VHstoredatam**.

This routine provides a high-level method for creating multi-order, single-field vdatas.

Note that there are two FORTRAN-77 versions of this routine; one for numeric data (**vhfsdm**) and the other for character data (**vhfscdm**).

FORTRAN

```
integer function vhfsdm(file_id, fieldname, buf, n_records,
                        data_type, vdata_name, vdata_class, order)

integer file_id, n_records, data_type, order

character*(*) vdata_name, vdata_class, fieldname
```

```
<valid numeric data type> buf(*)


integer function vhfscdm(file_id, fieldname, buf, n_records,
                        data_type, vdata_name, vdata_class
                        order)


integer file_id, n_records, data_type, order
character*(*) vdata_name, vdata_class, fieldname
character*(*) buf
```

VSappendable/vsapp (Obsolete)

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

vdata_id	IN:	Vdata identifier returned by **VSattach**
block_size	IN:	Standard block size of appended data

Purpose Makes it possible to append to a vdata.

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

Description The HDF library makes all vdatas appendable upon creation. Therefore, this routine has been made 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 of the vdata
access	IN:	Access mode

Purpose Attaches to an existing vdata or creates a new vdata.

Return value Returns a vdata identifier if successful and `FAIL` (or `-1`) otherwise.

Description **VSattach** attaches to the vdata identified by the reference number, *vdata_ref*, in the file identified by the parameter *file_id*. Access to the vdata is specified by the parameter *access*. **VSattach** returns an identifier to the vdata, through which all further operations on that vdata are carried out.

An existing vdata may be multiply-attached for reads. Only one attach with write access to a vdata is allowed.

The default interlace mode for a new vdata is `FULL_INTERLACE` (or `0`). This may be changed using **VSsetinterlace**.

The value of the parameter *vdata_ref* may be -1. This is used to create a new vdata.

Valid values for *access* are “`r`” for read access and “`w`” for write access.

If *access* is “`r`”, then *vdata_ref* must be the valid reference number of an existing vdata returned from any of the vdata and vgroup search routines (e.g., **Vgetnext** or **VSgetid**). It is an error to attach to a 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 or -1. An existing vdata is generally attached with “`w`” access to replace part of its data, or to append new data to it.

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,* intn *attr_index,* char **attr_name*, int32 **data_type*,
int32 **count*, int32 **size*)

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

Purpose Retrieves attribute information of a vdata or a vdata field.

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

Description **VSattrinfo** gets information on the attribute attached to the vdata, *vdata_id*, or to the vdata field. Vdata field is specified by its index, *field_index.* Attribute is specified by its index, *attr_index.* The attribute name is returned into the parameter *attr_name*, the data type is returned into the parameter *data_type*, the number of values of the attribute is returned into the parameter *count*, and the size of the attribute is returned into the parameter *size*.

The parameter *field_index* in **VSattrinfo** is the same as the parameter *field_index* in **VSsetattr**. It can be set to either an integer field index for the vdata field attribute, or `_HDF_VDATA` (or `-1`) to specify the vdata attribute.

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

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
```

VSdelete/vsfdlte

int32 VSdelete(int32 *file_id*, int32 *vdata_id*)

file_id	IN:	File identifier returned by **Hopen**
vdata_id	IN:	Vdata identifier returned by **VSattach**

Purpose Remove a vdata from a file.

Return value Returns `SUCCEED` (or `0`) if successful and `FAIL` (or `-1`) if not successful.

Description **VSdelete** removes the vdata identified by the parameter *vdata_id* from the file identified by the parameter *file_id*.

FORTRAN

```
integer function vsfdlte(file_id, vdata_id)


integer file_id, vdata_id
```

VSdetach/vsfdtch

int32 VSdetach(int32 *vdata_id*)

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

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

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

Description **VSdetach** detaches from the vdata identified by the parameter *vdata_id* and updates the vdata information in the file if there are any changes. All memory used for that vdata is freed.

The *vdata_id* identifier 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 identifier returned by **VSattach**

Purpose Determines the number of records in a vdata.

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

Description **VSelts** returns the number of records in the vdata identified by *vdata_id*.

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 identifier returned by **VSattach**
fieldname	IN:	Name of the 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** defines a field with the name specified by the parameter *fieldname*, of the data type specified by the parameter *data_type*, of the order specified by the parameter *order*, and within the vdata identified by the parameter *vdata_id*.

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 vdata fields may not be modified or deleted.

There are certain field names the HDF library recognizes as predefined. A list of these predefined field types can be found in the HDF User's Guide.

A field is defined by its name (*fieldname*), its type (*data_type*) and its order (*order*). A fieldname is any sequence of characters. By convention, fieldnames are usually a 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 data types listed in Table 1A in Section I of this manual.

The order of a field is the number of components in that field. A field containing the value of a simple variable, such a time or pressure, would have an order of 1. Compound variables have an order greater than 1. For example, a field containing the values associated with a variable for velocity in three dimensions would have an order of 3.

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 **field_name_list*)

vdata_id	IN:	Vdata identifier returned by **VSattach**
field_name_list	IN:	List of field names

Purpose Checks to see if certain fields exist in the current vdata.

Return value Returns a value of 1 if all field(s) exist and `FAIL` (or `-1`) otherwise.

Description **VSfexist** checks if all fields with the names specified in the parameter *field_name_list* exist in the vdata identified by the parameter *vdata_id.*

The parameter *field_name_list* is a string of comma-separated fieldnames (e.g., “PX,PY,PZ” in C and ’PX,PY,PZ’ in Fortran).

FORTRAN

```
integer function vsfex(vdata_id, field_name_list)


integer vdata_id

character*(*) field_name_list
```

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 Returns the reference number of a vdata, given its name.

Return value Returns the vdata reference number if successful and `0` if the vdata is not found or an error occurs.

Description **VSfind** returns the reference number of the vdata with the name specified by the parameter *vdata_name* in the file specified by the parameter *file_id*. 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
```

VSfindattr/vsffdat

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

vdata_id	IN:	Vdata identifier returned by **VSattach**
field_index	IN:	Field index
attr_name	IN:	Attribute name

Purpose Returns the index of an attribute of a vdata or vdata field.

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

Description **VSfindattr** returns the index of the attribute with the name specified by the parameter *attr_name* in the vdata identified by the parameter *vdata_id*.

To return the index of the attribute attached to the vdata , set the value of the parameter *field_index* to `_HDF_VDATA` (or `-1`). To return the index of the attribute of a field in the vdata , set the value of the parameter *field_index* to the field index. Valid values of *field_index* range from 0 to the total number of the vdata fields - 1. The number of the vdata fields is returned by **VFnfields**.

FORTRAN

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


integer vdata_id, field_index

character*(*) attr_name
```

VSfindclass/vffcls

int32 VSfindclass(int32 *file_id*, 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 with a given vdata class name

Return value Returns the reference number of the vdata if successful and `0` if the vdata is not found or an error occurs.

Description **VSfindclass** returns the reference number of the vdata with the class name specified by the parameter *vdata_class* in the file identified by the parameter *file_id*.

FORTRAN

```
integer function vffcls(vdata_id, vdata_class)


integer vdata_id

character*(*) vdata_class
```

VSfindex/vsffidx

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

vdata_id	IN:	Vdata identifier returned by **VSattach**
fieldname	IN:	Name of the field
field_index	OUT:	Index of the field

Purpose Retrieves the index of a field within a vdata.

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

Description **VSfindex** retrieves the index, *field_index*, of the field with a name specified by the parameter *fieldname*, within the vdata identified by the parameter *vdata_id*.

FORTRAN

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

integer vdata_id, field_index

character*(*) fieldname
```

VSfnattrs/vsffnas

int32 VSfnattrs (int32 *vdata_id*, int32 *field_index*)

vdata_id	IN:	Vdata identifier returned by **VSattach**
field_index	IN:	Index of the field

Purpose Returns the number of attributes attached to a vdata *or* the number of attributes attached to a vdata field.

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

Description **VSfnattrs** returns the number of attributes attached to a vdata specified by the parameter *vdata_id*, or the number of attributes attached to a vdata field, specified by the field index, *field_index*.

To return the number of attributes attached to the vdata , set the value of *field_index* to `_HDF_VDATA` (or `-1`). To return the number of attributes of a field in the vdata , set the value of *field_index* to the field index. Field index is a nonnegative integer less than the total number of the vdata fields. The number of vdata fields is returned by **VFnfields**.

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

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 **field_name_list*, VOIDP *bufptrs*[])

vdata_id	IN:	Vdata identifier returned by **VSattach**
action	IN:	Action to be performed
fields_in_buf	IN:	Names of the fields in *buf*
buf	IN/OUT:	Buffer containing the values of the packed fields to write to or read from the vdata
buf_size	IN:	Buffer size in bytes
n_records	IN:	Number of records to pack or unpack
field_name_list	IN:	Names of the fields to be packed or unpacked
bufptrs	IN/OUT:	Array of pointers to the field buffers

Purpose Packs field data into a buffer or unpacks buffered field data into vdata field(s) for fully interlaced fields.

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

Description **VSfpack** packs or unpacks the field(s) listed in the parameter *field_name_list* to or from the buffer *buf* according to the specified action in the parameter *action*.

Valid values for *action* are `_HDF_VSPACK` (or `0`) which packs field values from *bufptrs* (the field buffers) to *buf*, or `_HDF_VSUNPACK` (or `1`) which unpacks vdata field values from *buf* into *bufptrs*.

When **VSfpack** is called to pack field values into *buf*, *fields_in_buf* must list all fields of the vdata. When **VSfpack** is called to unpack field values, *fields_in_buf* may be a subset of the vdata fields. To specify all vdata fields in *fields_in_buf*, `NULL` can be used in C and a blank character (“ “) in Fortran.

The name(s) of the field(s) to be packed or unpacked are specified by the *field_name_list*. In C, the names in the parameter *field_name_list* can be a subset of or all field names listed in *fields_in_buf*. To specify all vdata fields, `NULL` can be used in C.

The FORTRAN-77 versions of this routine can pack or unpack only one field at a time. Therefore, *field_name_list* will contain the name of the field that will be packed or unpacked.

The calling program must allocate sufficient space for *buf* to hold all of the packed fields. The size of the *buf* buffer should be at least *n_records* * (the total size of all fields specified in *fields_in_buf*).

Note that there are two FORTRAN-77 versions of this routine: **vsfnpak** to pack or unpack a numeric field and **vsfcpak** to pack or unpack a character field.

Refer to the HDF User's Guide for an example on how to use this routine.

FORTRAN

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


integer vdata_id, action, buf(*), buf_size, n_records
<valid numeric data type> bufptrs(*)
character*(*) fields_in_buf, field_name_list
```

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


integer vdata_id, action, buf(*), buf_size, n_records

character*(*) fields_in_buf, field_name_list, bufptrs(*)
```

VSgetattr/vsfgnat/vsfgcat

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

vdata_id	IN:	Vdata identifier returned by **VSattach**
field_index	IN:	Index of the field
attr_index	IN:	Index of the attribute
values	OUT:	Buffer for the attribute values

Purpose Retrieves the attribute values of a vdata or vdata field.

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

Description **VSgetattr** retrieves the attribute values of the vdata identified by the parameter *vdata_id* or the vdata field specified by the field index, *field_index,* into the buffer *values*.

If *field_index* is set to `_HDF_VDATA` (or `-1`), the value of the attribute attached to the vdata is returned. If *field_index* is set to the field index, attribute attached to a vdata field is returned. Field index is a nonnegative integer less than the total number of the vdata fields. The number of vdata fields is returned by **VFnfields**

Attribute to be retrieved is specified by its index, *attr_index*. Index is a nonnegative integer less than the total number of the vdata or vdata field attributes. Use **VSfnattrs** to find the number of the vdata or vdata field attributes.

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, field_index, attr_index

character*(*) values
```

VSgetclass/vsfgcls

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

vdata_id	IN:	Vdata identifier returned by **VSattach**
vdata_class	OUT:	Vdata class name

Purpose Retrieves the vdata class name, if any.

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

Description **VSgetclass** retrieves the class name of the vdata identified by the parameter *vdata_id* and places it in the buffer *vdata_class*.

Space for the buffer *vdata_class* must be allocated by the calling program before **VSgetclass** is called. The maximum length of the class name is defined by the macro `VSNAMELENMAX (or 64)`.

FORTRAN

```
integer function vsfgcls(vdata_id, vdata_class)


integer vdata_id

character*(*) vdata_class
```

VSgetfields/vsfgfld

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

vdata_id	IN:	Vdata identifier returned by **VSattach**
field_name_list	OUT:	Field name list

Purpose Retrieves the field names of all of the fields in a vdata.

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

Description **VSgetfields** retrieves the names of the fields in the vdata identified by the parameter *vdata_id* into the buffer *field_name_list*.

The parameter *field_name_list* is a character string containing a comma-separated list of names (e.g., “PX,PY,PZ” in C or ‘PX,PY,PZ’ in Fortran).

FORTRAN

```
integer function vsfgfld(vdata_id, field_name_list)

integer vdata_id

character*(*) field_name_list
```

VSgetid/vsfgid

int32 VSgetid(int32 *file_id*, int32 *vdata_ref*)

file_id	IN:	File identifier returned by **Hopen**
vdata_ref	IN:	Vdata reference number

Purpose Sequentially searches through a file for vdatas.

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

Description **VSgetid** sequentially searches through a file identified by the parameter *file_id* 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. Searching past the last vdata in a file will result in an error condition.

To initiate a search, this routine must be called with the value of *vdata_ref* equal to -1. Doing so returns the reference number of the first vdata in the file.

FORTRAN

```
integer function vsfgid(file_id, vdata_ref)


integer file_id, vdata_ref
```

VSgetinterlace/vsfgint

int32 VSgetinterlace(int32 *vdata_id*)

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

Purpose Returns the interlace mode of a vdata.

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

Description **VSgetinterlace** returns the interlace mode of the vdata identified by the parameter *vdata_id*.

FORTRAN

```
integer function vsfgint(vdata_id)


integer vdata_id
```

VSgetname/vsfgnam

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

vdata_id	IN:	Vdata identifier returned by **VSattach**
vdata_name	OUT:	Vdata name

Purpose Retrieves the name of a vdata.

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

Description **VSgetname** retrieves the name of the vdata identified by the parameter *vdata_id* into the buffer *vdata_name*.

The user must allocate the memory space for the buffer *vdata_name* before calling **VSgetname**. If the vdata does not have a name, a null string is returned in the parameter *vdata_name* . The maximum length of a vdata name is defined by `VSNAMELENMAX` (or 64)

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 returned by **VSattach**

Purpose Returns the version number of a vdata.

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

Description **VSgetversion** returns the version number of the vdata identified by the parameter *vdata_id*. There are three valid version numbers: `VSET_OLD_VERSION` (or `2`), `VSET_VERSION` (or `3`), and `VSET_NEW_VERSION` (or `4`).

`VSET_OLD_VERSION` is returned when the vdata is of a version that corresponds to an HDF library version before version 3.2.

`VSET_VERSION` is returned when the vdata is of a version that corresponds to an HDF library version between versions 3.2 and 4.0 release 2.

`VSET_NEW_VERSION` is returned when the vdata is of the version that corresponds to an HDF library version of version 4.1 release 1 or higher.

FORTRAN

```
integer vsgver(vdata_id)


integer vdata_id
```

VSinquire/vsfinq

intn VSinquire(int32 *vdata_id*, int32 **n_records*, int32 **interlace_mode*, char **field_name_list*, int32 **vdata_size*, char **vdata_name*)

vdata_id	IN:	Vdata identifier returned by **VSattach**
n_records	OUT:	Number of records
interlace_mode	OUT:	Interlace mode of the data
field_name_list	OUT:	List of field names
vdata_size	OUT:	Size of a record
vdata_name	OUT:	Name of the vdata

Purpose Retrieves general information about a vdata.

Return value Returns `SUCCEED` (or `0`) if successful and `FAIL` (or `-1`) if it is unable to return any of the requested information.

Description **VSinquire** retrieves the number of records, the interlace mode of the data, the name of the fields, the size, and the name of the vdata, *vdata_id*, and stores them in the parameters *n_records*, *interlace_mode*, *field_name_list*, *vdata_size*, and *vdata_name*, respectively. In C, if any of the output parameters are `NULL`, the corresponding information will not be retrieved. Refer to the Reference Manual pages on **VSelts**, **VSgetfields**, **VSgetinterlace**, **VSsizeof** and **VSgetname** for other routines that can be used to retrieve specific information.

Possible returned values for *interlace_mode* are `FULL_INTERLACE` (or `0`) and `NO_INTERLACE` (or `1`). The returned value of *vdata_size* is the number of bytes in a record and is machine-dependent.

The parameter *field_name_list* is a character string that contains the names of all the vdata fields, separated by commas. (e.g., “PX,PY,PZ” in C and ’PX,PY,PZ’ in Fortran).

FORTRAN

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


integer vdata_id, n_records, interlace, vdata_size

character*(*) field_name_list, vdata_name
```

VSisattr/vsfisat

intn VSisattr(int32 *vdata_id*)

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

Purpose Determines whether a vdata is an attribute.

Return value Returns `TRUE` (or `1`) if the vdata is an attribute, and `FALSE` (or `0`) otherwise.

Description **VSisattr** determines whether the vdata identified by the parameter *vdata_id* is an attribute.

As attributes are stored by the HDF library as vdatas, a means of testing whether or not a particular vdata is an attribute is needed, and is provided by this 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:	Array of reference numbers
max_refs	IN:	Maximum number of lone vdatas to be retrieved

Purpose Retrieves the reference numbers of all lone vdatas, i.e., vdatas that are not grouped with other objects, in a file.

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

Description **VSlone** retrieves the reference numbers of lone vgroups in the file identified by the parameter *file_id*. Although **VSlone** returns the number of lone vdatas in the file, only at most *max_refs* reference numbers are retrieved and stored in the buffer *ref_array.* The array must have at least *max_refs* elements.

An array size of 65,000 integers for *ref_array* is more than adequate if the user chooses to declare the array statically. However, the preferred method is to dynamically allocate memory instead; first call **VSlone** with a value of 0 for *max_refs* to return the total number of lone vdatas, then use the returned value to allocate memory for *ref_array* before calling **VSlone** again.

FORTRAN

```
integer function vsflone(file_id, ref_array, max_refs)

integer file_id, ref_array(*), max_refs
```

VSnattrs/vsfnats

intn VSnattrs(int32 *vdata_id*)

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

Purpose Returns the total number of attributes of a vdata and of its fields.

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

Description **VSnattrs** returns the total number of attributes of the vdata, *vdata_id*, and of its fields.

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

FORTRAN

```
integer function vsfnats(vdata_id)


integer vdata_id
```

VSread/vsfrd/vsfrdc/vsfread

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

vdata_id	IN:	Vdata identifier returned by **VSattach**
databuf	OUT:	Buffer to store the retrieved data
n_records	IN:	Number of records to be retrieved
interlace_mode	IN:	Interlace mode of the data to be stored in the buffer

Purpose Retrieves data from a vdata.

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

Description **VSread** reads *n_records* records from the vdata identified by the parameter *vdata_id* and stores the data in the buffer *databuf* using the interlace mode specified by the parameter *interlace_mode*.

The user can specify the fields and the order in which they are to be read by calling **VSsetfields** prior to reading. **VSread** stores the requested fields in *databuf* in the specified order.

Valid values for *interlace_mode* are `FULL_INTERLACE` (or `1`) and `NO_INTERLACE` (or `0`). Selecting `FULL_INTERLACE` causes *databuf* to be filled by record and is recommended for speed and efficiency. Specifying `NO_INTERLACE` causes *databuf* to be filled by field, i.e., all values of a field in *n_records* records are filled before moving to the next field. Note that the default interlace mode of the buffer is `FULL_INTERLACE`.

As the data is stored contiguously in the vdata, **VSfpack** should be used to unpack the fields after reading. Refer to the discussion of **VSfpack** in the HDF User's Guide for more information.

Note that there are three FORTRAN-77 versions of this routine: **vsfrd** is for buffered numeric data, **vsfrdc** is for buffered character data and **vsfread** is for generic packed data.

FORTRAN

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


integer vdata_id, n_records, interlace_mode

<valid numeric data type> databuf(*)




integer function vsfrdc(vdata_id, databuf, n_records,
                            interlace_mode)
```

```
integer vdata_id, n_records, interlace_mode

character*(*) databuf


integer function vsfread(vdata_id, databuf, n_records,
                            interlace_mode)


integer vdata_id, n_records, interlace_mode

integer databuf(*)
```

VSseek/vsfseek

int32 VSseek(int32 *vdata_id*, int32 *record_pos*)

vdata_id	IN:	Vdata identifier returned by **VSattach**
record_pos	IN:	Position of the record

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 **VSseek** moves the access pointer within the vdata identified by the parameter *vdata_id* to the position of the record specified by the parameter *record_pos*. The next call to **VSread** or **VSwrite** will read from or write to the record where the access pointer has been moved to.

The value of *record_pos* is zero-based. For example, to seek to the third record in the vdata, set *record_pos* to 2. The first record position is specified by specifying a *record_pos* value of 0. Each seek is constrained to a record boundary within the vdata.

FORTRAN

```
integer function vsfseek(vdata_id, record_pos)


integer vdata_id, record_pos
```

VSsetattr/vsfsnat/vsfscat

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

vdata_id	IN:	Vdata identifier returned by **VSattach**
field_index	IN:	Index of the field
attr_name	IN:	Name of the attribute
data_type	IN:	Data type of the attribute
count	IN:	Number of attribute values
values	IN:	Buffer containing the attribute values

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

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

Description **VSsetattr** defines an attribute that has the name specified by the parameter *attr_name*, the data type specified by the parameter *data_type*, and the number of values specified by the parameter *count*, and that contains the values specified in the parameter *values*. The attribute is set for either the vdata or a vdata field depending on the value of the parameter *field_index*.

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 a value of `FAIL` (or `-1`) to be returned.

If *field_index* value is set to `_HDF_VDATA` (or `-1`), the attribute will be set for the vdata. If *field_index* is set to the field index, attribute will be set for the vdata field. Field index is a nonnegative integer less than the total number of the vdata fields. The number of vdata fields can be obtained using **VFnfields**.

The value of the parameter *data_type* can be any one of the data types listed in Table 1A in Section I of this manual.

FORTRAN

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


integer vdata_id, field_index, data_type, count, values(*)

character*(*) attr_name




integer function vsfscat(vdata_id, field_index, attr_name,
                               data_type, count, values)
```

```
integer vdata_id, field_index, data_type, count

character*(*) attr_name, values(*)
```

VSsetclass/vsfscls

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

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

Purpose Sets the class name of a vdata.

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

Description **VSsetclass** sets the class name of the vdata identified by the parameter *vdata_id* to the value of the parameter *vdata_class*.

At creation, the class name of a vdata is `NULL`. The class name may be reset more than once. Class names, like vdata names, can be any character string. They exist solely as meaningful labels to user applications and are not used by the HDF library in any way. Class names will be truncated to `VSNAMELENMAX` (or `64`) characters.

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 by **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 **VSsetexternalfile** writes data in the vdata identified by the parameter *vdata_id* in the file named *filename*, at the byte offset specified by the parameter *offset*.

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

IMPORTANT: The user 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 **field_name_list*)

vdata_id	IN:	Vdata identifier returned by **VSattach**
field_name_list	IN:	List of the field names to be accessed

Purpose Specifies the fields to be accessed.

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

Description **VSsetfields** specifies that the fields, whose names are listed in the parameter *field_name_list*, of the vdata identified by the parameter *vdata_id* will be accessed by the next call to **VSread** or **VSwrite**. **VSsetfields** must be called before any call to **VSread** or **VSwrite**.

For reading from a vdata, a call to **VSsetfields** sets up the fields that are to be retrieved from the records in the vdata. If the vdata is empty, **VSsetfields** will return `FAIL` (or `-1`).

For writing to a vdata, **VSsetfields** can only be called once, to set up the fields in a vdata. Once the vdata fields are set, they may not be changed. Thus, to update some fields of a record after the first write, the user must read all the fields to a buffer, update the buffer, then write the entire record back to the vdata.

The parameter *field_name_list* is a character string that contains a comma-separated list of fieldnames (i.e., “PX,PY,PZ” in C and ’PX,PY,PZ’ in Fortran). The combined width of the fields in a vdata must be less than `MAX_FIELD_SIZE` (or `65535`) bytes. If an attempt to create a larger record is made, **VSsetfields** will return `FAIL` (or `-1`).

If the vdata is attached with an “`r`” access mode, the parameter *field_name_list* must contain only the fields that already exist in the vdata. If the vdata is attached with a “`w`” access mode, *field_name_list* can contain the names of any fields that have been defined by **VSfdefine** or any predefined fields.

FORTRAN

```
integer function vsfsfld(vdata_id, field_name_list)

integer vdata_id

character*(*) field_name_list
```

VSsetinterlace/vsfsint

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

vdata_id	IN:	Vdata identifier returned by **VSattach**
interlace_mode	IN:	Interlace mode of the data to be stored in the vdata

Purpose Sets the interlace mode of a vdata.

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

Description **VSsetinterlace** sets the interlace mode of the vdata, *vdata_id*, to that specified by the parameter *interlace_mode*. This routine can only be used when creating new vdatas with write access.

The value of *interlace_mode* may be either `FULL_INTERLACE` (or `0`) or `NO_INTERLACE` (or `1`). If this routine is not called, the default interlace mode of the vdata is `FULL_INTERLACE`. The `FULL_INTERLACE` option is more efficient than `NO_INTERLACE` although both require the same amount of disk space.

Specifying `FULL_INTERLACE` accesses the vdata by record; in other words, all values of all fields in a record are accessed before moving to the next record. Specifying `NO_INTERLACE` accesses the vdata by field; in other words, all field values are accessed before moving to the next field. Thus, for writing data, all record data must be available before the write operation is invoked.

Note that the interlace mode of the data to be written is specified by a parameter of the **VSwrite** routine.

FORTRAN

```
integer function vsfsint(vdata_id, interlace_mode)


integer vdata_id, interlace_mode
```

VSsetname/vsfsnam

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

vdata_id	IN:	Vdata identifier returned by **VSattach**
vdata_name	IN:	Name of the vdata

Purpose Assigns a name to a vdata.

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

Description **VSsetname** sets the name of the vdata identified by the parameter *vdata_id* to the value of the parameter *vdata_name*.

At creation, the name of the vdata is `NULL`. The name may be reset more than once. Vdata names, like class names, can be any character string. They exist solely as a meaningful label for user applications and are not used by the HDF library in any way. Vdata names will be truncated to `VSNAMELENMAX` (or `64`) characters.

FORTRAN

```
integer function vsfsnam(vdata_id, vdata_name)


integer vdata_id

character*(*) vdata_name
```

VSsizeof/vsfsiz

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

vdata_id	IN:	Vdata identifier returned by **VSattach**
field_name_list	IN:	Name(s) of the fields to check

Purpose Computes the size, in bytes, of the given field(s) for the local machine.

Return value Returns the fields size if successful and `FAIL` (or `-1`) otherwise.

Description **VSsizeof** computes the size, in bytes, of the fields specified in the parameter *field_name_list* in the vdata identified by the parameter *vdata_id*.

The parameter *field_name_list* specifies a single field or several comma-separated fields. The field or fields should already exist in the vdata. If more than one field is specified, **VSsizeof** will return the total sizes of all of the fields.

FORTRAN

```
integer function vsfsiz(vdata_id, field_name_list)


integer vdata_id

character*(*) field_name_list
```

VSwrite/vsfwrt/vsfwrtc/vsfwrit

int32 VSwrite(int32 *vdata_id*, uint8 **databuf*, int32 *n_records*, int32 *interlace_mode*)

vdata_id	IN:	Vdata identifier returned by **VSattach**
databuf	IN:	Buffer of records to be written to the vdata
n_records	IN:	Number of records to be written
interlace_mode	IN:	Interlace mode of the buffer in memory

Purpose Writes data to a vdata.

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

Description **VSwrite** writes the data stored in the buffer *databuf* into the vdata identified by the parameter *vdata_id*. The parameter *n_records* specifies the number of records to be written. The parameter *interlace_mode* defines the interlace mode of the vdata fields stored in the buffer *databuf.*

Valid values for *interlace_mode* are `FULL_INTERLACE` (or `0`) and `NO_INTERLACE` (or `1`). Selecting `FULL_INTERLACE` fills *databuf* by record and is recommended for speed and efficiency. Specifying `NO_INTERLACE` causes *databuf* to be filled by field, i.e., all values of a field in all records must be written before moving to the next field. Thus, all data must be available before writing. If the data is to be written to the vdata with an interlace mode different from that of the buffer, **VSsetinterlace** must be called prior to **VSwrite**. Note that the default interlace mode of a vdata is `FULL_INTERLACE`.

It is assumed that the data in *databuf* is organized as specified by the parameter *interlace_mode*. The number and order of the fields organized in the buffer must correspond with the number and order of the fields specified in the call to **VSsetfields**, which finalizes the vdata fields definition. Since **VSwrite** writes the data in *databuf* contiguously to the vdata, **VSfpack** must be used to remove any “padding”, or non-data spaces, used for vdata field alignment. This process is called packing. Refer to the discussion of **VSfpack** in the HDF User’s Guide for more information.

Before writing data to a newly-created vdata, **VSdefine** and **VSsetfields** must be called to define the fields to be written.

Note that there are three FORTRAN-77 versions of this routine: **vsfwrt** is for buffered numeric data, **vsfwrtc** is for buffered character data and **vsfwrit** is for generic packed data.

FORTRAN

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


integer vdata_id, n_records, interlace_mode

<valid numeric data type> databuf(*)




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


integer vdata_id, n_records, interlace_mode

character*(*) databuf


integer function vsfwrit(vdata_id, databuf, n_records,
                                interlace_mode)


integer vdata_id, n_records, interlace_mode

character*(*) databuf
```