VHstoredata/vhfsd/vhfscd

int32 VHstoredata(int32 *file\_id*, char \**fieldname*, uint8 *buf*[], int32 *n\_records*, int32 *ntype*, 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 |
| ntype | 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 number type are specified by the parameters vdata\_name, vdata\_class, and ntype, 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, ntype, vdata\_name, vdata\_class) | | |
|  | integer file\_id, n\_records, ntype | | |
|  | character\*(\*) vdata\_name, vdata\_class, fieldname | | |
|  | <valid numeric data type> buf(\*) | | |
|  |  | | |
|  | integer function vhfscd(file\_id, fieldname, buf, n\_records, ntype, vdata\_name, vdata\_class) | | |
|  | integer file\_id, n\_records, ntype | | |
|  | character\*(\*) vdata\_name, vdata\_class, fieldname | | |
|  | character\*(\*) buf | | |

VHstoredatam/vhfsdm/vhfscdm

int32 VHstoredatam(int32 *file\_id*, char \**fieldname*, uint8 *buf*[], int32 *n\_records*, int32 *ntype*, 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 |
| ntype | 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 number type of the vdata is specified by the parameter ntype. 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, | | |
|  | integer file\_id, n\_records, ntype, order | | |
|  | character\*(\*) vdata\_name, vdata\_class, fieldname | | |
|  | <valid numeric data type> buf(\*) | | |
|  |  | | |
|  | integer function vhfscdm(file\_id, fieldname, buf, n\_records, ntype, vdata\_name, vdata\_class, order) | | |
|  | integer file\_id, n\_records, ntype, 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 \**ntype,* 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 |
| ntype | OUT: | Number 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 number type is returned into the parameter ntype, 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, ntype, 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, ntype, count, size) | | |
|  | integer vdata\_id, field\_index, attr\_index | | |
|  | character\*(\*) attr\_name | | |
|  | integer ntype, count, size | | |

VSdelete/vsfdlte

int32 VSdelete(int32 *file\_id*, int32 *vdata\_ref*)

|  |  |  |
| --- | --- | --- |
| file\_id | IN: | File identifier returned by Hopen |
| vdata\_ref | IN: | Vdata reference number 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\_ref from the file identified by the parameter file\_id. | | |
| FORTRAN | integer function vsfdlte(file\_id, vdata\_ref) | | |
|  | integer file\_id, vdata\_ref | | |

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 *ntype*, int32 *order*)

|  |  |  |
| --- | --- | --- |
| vdata\_id | IN: | Vdata identifier returned by VSattach |
| fieldname | IN: | Name of the field to be defined |
| ntype | IN: | Number 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 number type specified by the parameter ntype, 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 (ntype) 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, ntype may be one of the number 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, ntype, order) | | |
|  | integer vdata\_id, ntype, 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 | | |
|  | character\*(\*) fields\_in\_buf, field\_name\_list | | |
|  | <valid numeric data type> bufptrs(\*) | | |
|  |  | | |
|  | 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(\*) | | |

VSgetattdatainfo

intn VSgetattdatainfo(int32 *vdata\_id*, int32 *field\_index*, char\* *attr\_index*, int32\* *offset*, int32\* *length*)

|  |  |  |
| --- | --- | --- |
| vdata\_id | IN: | Vdata identifier returned by VSattach |
| field\_index | IN: | Index of the field |
| attr\_index | IN: | Index of the attribute |
| offset | OUT: | Offset of the attribute’s data |
| length | OUT: | Length of the attribute’s data |

|  |  |
| --- | --- |
| Purpose | Retrieves location and size of the data of an attribute. |
| Return value | Returns the number of data blocks retrieved, which should be 1, if successful and FAIL (or -1) otherwise. |
| Description | VSgetattdatainfo retrieves the offset and length of the data that belongs to the attribute specified its index, attr\_index. The specified attribute is either attached to the vdata, specified by vdata\_id, or to the vdata field, depending on the value of the parameter field\_index. To specify an attribute of a vdata, the application will set field\_index to \_HDF\_VDATA (or -1). To specify an attribute of a vdata field, the application will set field\_index to the index of the vdata field. A valid 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 parameter attr\_index specifies the position of the attribute in the list of all attributes belonging to the vdata or the vdata field. VSfnattrs routine can be used to obtain the number of attributes of a vdata or of a field contained in the vdata. |

|  |  |
| --- | --- |
| FORTRAN | Currently unavailable |
|  |  |

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 | | |

VSgetblockinfo/vsfgetblinfo

intn VSgetblockinfo(int32 *vdata\_id*, int32 \**block\_size*, int32 \**num\_blocks*)

|  |  |  |
| --- | --- | --- |
| vdata\_id | IN: | Vdata identifier |
| block\_size | OUT: | Block size in bytes |
| num\_blocks | OUT: | Number of linked blocks |
| Purpose | Retrieves the block size and the number of blocks for a linked-block vdata element. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | **VSgetblockinfo** retrieves the block size and the number of linked blocks for a linked-block vdata element. | | |
|  | If no response is desired for either returned value, block\_size and num\_blocks may be set to NULL. | | |
| FORTRAN | integer function vsfgetblinfo(vdata\_id, block\_size, num\_blocks) | | |
|  | integer vdata\_id, num\_blocks, block\_size | | |
|  |  | | |

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 | | |

VSgetdatainfo

intn VSgetdatainfo(int32 *vdata\_id*, uintn *start\_block*, uintn *info\_count*, int32 \**offsetarray*, int32 \**lengtharray*)

|  |  |  |
| --- | --- | --- |
| vdata\_id | IN: | Vdata identifier returned by VSattach |
| start\_block | IN: | Value indicating where to start reading offsets |
| info\_count | IN: | Number of elements each offset or length list can hold |
| offsetarray | OUT: | Array to hold offsets of the data blocks |
| lengtharray | OUT: | Array to hold lengths of the data blocks |
| Purpose | Retrieves location and size of data blocks in a specified vdata, after a specified data block. | | |
| Return value | Returns the actual number of blocks in the vdata's data or the number of blocks retrieved if successful and FAIL (or -1) otherwise. | | |
| Description | VSgetdatainfo retrieves two lists containing the offsets and lengths of the blocks of data belonging to the vdata specified by vdata\_id. | | |
|  | The parameter info\_count provides the number of offset/length values that the lists offsetarray and lengtharray can hold. The application can first invoke VSgetdatainfo passing in 0 for info\_count and NULL for both arrays to get the value for info\_count and to provide proper memory allocation for offsetarray and lengtharray in the next call to VSgetdatainfo. | | |
|  | The parameter start\_block indicates the block number where to start reading the offsets from the file. The combination of parameters info\_count and start\_block provide applications with flexibility of where and how much data information to retrieve. The value for start\_block must be non-negative and smaller than or equal to the number of blocks in the vdata's data.  When start\_block is 0, VSgetdatainfo will start getting data info from the beginning of the vdata's data.  When start\_block is greater than the number of blocks in the vdata, VSgetdatainfo will return FAIL (or -1). | | |
| FORTRAN | currently unavailable | | |
|  |  | | |
|  |  | | |

VSgetexternalinfo

intn VSgetexternalinfo(int32 *vdata\_id,* uintn *buf\_size*, char \**filename*, int32 \**offset*, int32 \**length*)

|  |  |  |
| --- | --- | --- |
| vdata\_id | IN: | Vdata identifier returned by VSattach |
| buf\_size | IN: | Size of buffer for external file name |
| filename | OUT: | Buffer for external file name |
| offset | OUT: | Offset, in bytes, of the location in the external file where the data was written |
| length | OUT: | Length, in bytes, of the external data |
| Purpose | Retrieves information about external file and external data of the vdata. | | |
| Return value | Returns length of the external file name if successful, 0 if there is no external data, or FAIL (or -1) if an error occurs. | | |
| Description | If the vdata has external element, VSgetexternalinfo will retrieve the name of the external file, the offset where the data is being stored in the external file, and the length of the external data. If the vdata does not have external element, VSgetexternalinfo will return 0. | | |
|  | To sufficiently allocate buffer for the file name, an application can call VSgetexternalinfo passing in 0 for buf\_size. If the length returned is greater than 0, the application will use it to allocate the buffer before calling VSgetexternalinfo again to get the actual file name. | | |
| Note | It is the user's responsibility to see that the external files are kept with the main file prior to accessing the vdata with external element. VSgetexternalinfo does not check and the accessing functions will fail if the external file is missing from the directory where the main file is located. | | |
| FORTRAN | Currently unavailable | | |
|  |  | | |

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). | | |
|  | The caller must allocate adequate memory for the buffer field\_name\_list before calling VSgetfields. The combined width of the fields in a vdata is less than MAX\_FIELD\_SIZE (or 65535.) | | |
|  | If the vdata does not have any fields, a null string is returned in the parameter field\_name\_list. | | |
| 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 | | |

VSgetvdatas/vsfgvdatas

intn VSgetvdatas(int32 *id*, const uintn *start\_vd*, const uintn *n\_vds*, uint16 \**refarray*)

|  |  |  |
| --- | --- | --- |
| id | IN: | File identifier returned by Hopen or vgroup identifier returned by Vattach |
| start\_vd | IN: | Vdata number to start retrieving at |
| vd\_count | IN: | Number of vdatas to be retrieved |
| refarray | OUT: | Array to hold reference numbers of retrieved vdatas |
| Purpose | Retrieves reference numbers of vdatas in a file or in a vgroup. | | |
| Return value | Returns the actual number of user-created vdatas retrieved if successful, and FAIL (-1) otherwise. | | |
| Description | VSgetvdatas retrieves a list containing the reference numbers of vdatas found in a file or a vgroup. The file or the vgroup is specified by id. | | |
|  | The retrieved vdatas will be the ones that were previously created by user applications, not including those that were created by the library internally. They are referred to as user-created vdatas, for brevity. | | |
|  | The parameter vd\_count provides the number of items that the list refarray can hold. The retrieval starts at the vdata number start\_vd going forward in the order which the vdatas were created. For example, if there are 100 vdatas that can be retrieved, specifying start\_vd as 90 and vd\_count as 10 will retrieve the last ten vdatas. The value for start\_vd must be non-negative and smaller than or equal to the number of user-created vdatas in the specified file or vgroup. | | |
|  | When start\_vd is 0, VSgetvdatas will start retrieving at the beginning of the file or the first vdata of the specified vgroup.  When start\_vd is between 0 and the number of user-created vdatas in the file or the vgroup, VSgetvdatas will start retrieving vdatas from the vdata number start\_vd.  When start\_vd is greater than the number of user-created vdatas in the file or the vgroup, VSgetvdatas will return FAIL. | | |
|  | To allocate sufficient memory for refarray, the application can first invoke VSgetvdatas passing in NULL for refarray to get the value for vd\_count then call VSgetvdatas again with proper memory allocation for refarray. | | |
|  | When id is a vgroup identifier, only the immediate vdatas will be retrieved; that is, the sub-vgroups will not be searched. | | |
| FORTRAN | integer function vsfgvdatas(id, start\_vd, vd\_count, refarray) | | |
|  | integer id, start\_vd, vd\_count | | |
|  | integer refarray(\*) | | |

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). | | |
|  | The user must allocate the memory space for the buffer vdata\_name before calling VSinquire. 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) | | |
| Note | VSinquire will return FAIL if it is called before VSdefine and VSsetfield on the same vdata. | | |
| 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 | | |

VSisinternal

intn VSisinternal(int32 *vdata\_id*)

|  |  |  |
| --- | --- | --- |
| vdata\_id | IN: | Vdata identifier returned by VSattach |
| Purpose | Determine if a vdata was created by the library internally. | | |
| Return value | Returns TRUE (1) if the inquired vdata is one that was internally created by the library, FALSE (0) otherwise, and FAIL (-1) if failure occurs. | | |
| Description | VSisinternal checks the class name of the given vdata against the list HDF\_INTERNAL\_VDS to determine whether the vdata was previously created by the library instead of by a user application. | | |
|  | The names in HDF\_INTERNAL\_VDS are:  DIM\_VALS ("DimVal0.0")  DIM\_VALS01 ("DimVal0.1")  \_HDF\_ATTRIBUTE ("Attr0.0")  HDF\_SDSVAR ("SDSVar")  HDF\_CRDVAR ("CoordVar")  \_HDF\_CHK\_TBL\_CLASS ("\_HDF\_CHK\_TBL\_")  RIGATTRCLASS("RIATTR0.0C") | | |
| FORTRAN | Currently unavailable | | |
|  |  | | |

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 vdatas 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 | | |

VSofclass

intn VSofclass(int32 *id*, const char \**vsclass*, const uintn *start\_vd*, const uintn *n\_vds*, uint16 \**refarray*)

|  |  |  |
| --- | --- | --- |
| id | IN: | File identifier, returned by Hopen, or vgroup identifier, returned by Vattach |
| vsclass | IN: | Name of class for vdatas to be queried |
| start\_vd | IN: | Vdata number to start retrieving at |
| n\_vds | IN: | Number of vdatas to retrieve |
| refarray | OUT: | Array to hold vdata reference numbers |
| Purpose | Retrieves reference numbers of vdatas of the specified class. | | |
| Return value | Returns 0 if none is found, FAIL(-1) if error occurs, or the number of reference numbers returned in the refarray, if successful. | | |
| Description | VSofclass retrieves n\_vds vdatas by their reference numbers via the caller-supplied array refarray. The vdatas to be retrieved have class name as vsclass. | | |
|  | The parameter n\_vds provides the number of values that the refarray list can hold and can be any positive number smaller than MAX\_REF (65535). If n\_vds is larger than the actual number of vdatas that has the specified class, then only the actual number of vdatas will be retrieved. | | |
|  | The parameter start\_vd specifies the vdata number where the retrieval will start.  When start\_vd is 0, VSofclass will start retrieving at the beginning.  When start\_vd is between 0 and the number of vdatas that meet the search criteria, VSofclass will start retrieving from the vdata number start\_vd.  When start\_vd is greater than the number of vdatas that meet the search criteria, VSofclass will return FAIL. | | |
|  | When refarray argument is NULL, VSofclass will return the actual number of vdatas that meet the search criteria. This will allow application to determine the size of the array for dynamic allocation before invoking VSofclass again. | | |
| FORTRAN | Currently unavailable | | |
|  |  | | |

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. | | |
|  | See the notes regarding the potential performance impact of appendable data sets in the *HDF User’s Guide* Section 14.4.3, "Unlimited Dimension Data Sets (SDSs and Vdatas) and Performance." | | |
| FORTRAN | On Windows systems, this function is available only for an integer data buffer. | | |
|  | 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. | | |
|  | See the notes regarding the potential performance impact of appendable data sets in the HDF User’s Guide Section 14.4.3, "Unlimited Dimension Data Sets (SDSs and Vdatas) and Performance." | | |
| 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 *ntype,* 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 |
| ntype | IN: | Number 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 number type specified by the parameter ntype, 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 number type and order are the same as the current ones. Any changes in the field number 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 ntype can be any one of the number types listed in Table 1A in Section I of this manual. | | |
| FORTRAN | integer function vsfsnat(vdata\_id, field\_index, attr\_name, ntype, count, values) | | |
|  | integer vdata\_id, field\_index, ntype, count, values(\*) | | |
|  | character\*(\*) attr\_name | | |
|  |  | | |
|  | integer function vsfscat(vdata\_id, field\_index, attr\_name, ntype, count, values) | | |
|  | integer vdata\_id, field\_index, ntype, count | | |
|  | character\*(\*) attr\_name, values(\*) | | |

VSsetblocksize/vsfsetblsz

intn VSsetblocksize(int32 *vdata\_id*, int32 *block\_size*)

|  |  |  |
| --- | --- | --- |
| vdata\_id | IN: | Vdata identifier |
| block\_size | IN: | Size of each block in bytes |
| Purpose | Sets linked-block Vdata element block size. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | **VSsetblocksize** sets the block size for linked-block elements that will be used to store Vdatas. | | |
|  | The default block size is HDF\_APPENDABLE\_BLOCK\_LEN, which is set to 4096 in the library as it is distributed. **VSsetblocksize** modifies that default value and must be called before the first write to the Vdata. Once the linked-block element is created, the block size cannot be changed. | | |
|  | The following note may be of interest to users who must pay very close attention to performance issues: **VSsetblocksize** sets the block size only for blocks following the first block. The first block can be arbitrarily large; the library continues to write to it until it encounters an obstacle, at which point the linked block mechanism is invoked. For example, a Vdata A that is the last item in a file can continue to grow, simply extending the file. If a new Vdata B is then written, that new object is (normally) placed at the end of the file, blocking off extension of the prior Vdata, A. At this point, new writes to A will write data to linked blocks per the block\_size and num\_blocks settings. | | |
| FORTRAN | integer function vsfsetblsz(vdata\_id, block\_size) | | |
|  | integer vdata\_id, block\_size | | |
|  |  | | |

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. Consequently, the library does not check for uniqueness among vdatas. In addition, 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. | | |
|  | Refer to 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. Consequently, the library does not check for uniqueness of the name. In addition, vdata names will be truncated to VSNAMELENMAX (or 64) characters. | | |
| FORTRAN | integer function vsfsnam(vdata\_id, vdata\_name) | | |
|  | integer vdata\_id | | |
|  | character\*(\*) vdata\_name | | |

VSsetnumblocks/vsfsetnmbl

intn VSsetnumblocks(int32 *vdata\_id*, int32 *num\_blocks*)

|  |  |  |
| --- | --- | --- |
| vdata\_id | IN: | Vdata identifier |
| num\_blocks | IN: | Number of blocks to be used for the linked-block element |
| Purpose | Sets the number of blocks for a linked-block Vdata element. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | **VSsetnumblocks** sets the number of blocks in linked-block elements that will be used to store Vdatas. | | |
|  | The default number of blocks is HDF\_APPENDABLE\_BLOCK\_NUM, which is set to 16 in the library as it is distributed. **VSsetnumblocks** modifies that default value and must be called before the first write to the Vdata. Once the linked-block element is created, the number of blocks cannot be changed. | | |
| FORTRAN | integer function vsfsetnmbl(vdata\_id, num\_blocks) | | |
|  | integer vdata\_id, num\_blocks | | |
|  |  | | |

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 | On Windows systems, this function is available only for an integer data buffer. | | |
|  | 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 | | |