# *The VAPOR IDL Extension Reference Manual*

March, 2009

Version 1.4.4




National Center for Atmospheric Research

# Introduction

This reference is a complete listing of all extensions to IDL's built-in functions and procedures provided by VAPOR. The majority of the IDL language extensions provided by VAPOR operate on VAPOR Data Collections (VDCs). An overview of the VDC is provided in the document *An Overview of VAPOR Data Collections*. All functions and procedures are listed alphabetically. A description of each routine follows the name.

Conventions used in this guide are similar to those used by the *IDL Reference Guide*.

## *Common Arguments*

Many of the routines in this document accept a common set of arguments. These common arguments are described below.

### dfd

A VDC file handle. Almost all of the IDL routines with a *VDC_* prefix require a dfd as their first argument. The exceptions to this are the read and write objects that create and return a dfd: VDC_BUFREADCREATE, VDC_BUFWRITECREATE, VDC_REGREADCREATE, VDC_REGWRITECREATE. The dfd handle binds operations to a specific VDC.

### mfd

A metadata file handle. Almost all of the IDL routines with a *VDF_* prefix require an *mfd* as their first argument. The exception to this is the VDF_CREATE function which returns a *mfd*. The mfd handle binds operations to a VDC's metadata.

### reflevel

The integer refinement level for an operation. VDC data volumes may exist at multiple, discrete refinement levels. Each refinement level contains approximately 2x as many voxels along each coordinate dimension. The refinement level zero (0) always refers to the coarsest approximation in the VDC. Level one refers to the next coarsest, and so on. Unless otherwise noted a refinement level of -1 may be used to indicate the native resolution of the volume.

### tag

A user-defined keyword.

### timestep

An integer identifying the time step of a volume within a VDC. The first time step is zero.

### varname

A character string identifying the name of a variable within a VDC.

### vdffile

The path name to the metafile associated with a VDC.

# VDC_BUFREADCREATE

This function creates a buffered read object for reading data from a VDC one slice at a time.

## *Syntax*

```
dfd = VDC_BUFREADCREATE(mfd)
dfd = VDC_BUFREADCREATE(vdffile)
```

## *Return Value*

The function returns a handle which may be used for subsequent buffered read operations.

## *Arguments*

### mfd

A VDC metafile data handle (see the introduction).

### vdffile

The path to a VDC metafile (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDC_OPENVARREAD, VDC_CLOSEVAR, VDC_BUFREADSLICE, VDC_BUFREADDESTROY

# VDC_BUFREADDESTROY

This procedure destroys a buffered read object previously created with VDC_BUFREADCREATE

## *Syntax*

```
VDC_BUFREADDESTROY, dfd
```

## *Arguments*

### dfd

A VDC data handle (see the introduction).

## *Keywords*

## *Examples*

## *See Also*

VDC_BUFREADCREATE

# VDC_BUFREADSLICE

This procedure reads a single slice (2D array) of floating point data from a VDC. The VDC handle argument, *dfd*, determines the variable, time-step, and refinement level for the read based on the currently opened variable (See VDC_OPENVARREAD). VDC_BUFREADSLICE may be called repeatedly until the entire variable volume is read. The handle, *dfd*, must have been returned from a prior call to VDC_BUFREADCREATE.

## *Syntax*

```
VDC_BUFREADSLICE, dfd, slice
```

## *Arguments*

### dfd

A VDC data handle (see the introduction).

### Slice

A 2D floating point array of dimension Nx by Ny, where Nx and Ny are the X and Y coordinate dimensions of the volume (in voxels) at the desired refinement level associated with the opened VDC handle, *dfd*.

## *Keywords*

## *Examples*

The example below shows how to read a data volume from a VDC using the VDC_BUFREADSLICE procedure.

```
; The refinement level we’re interested in
;
reflevel = 1

;
; Create a VDF metadata object from an existing
; metadata file. The metadata file must already
; exist on disk
vdffile = 'test.vdf'
mfd = vdf_create(vdffile)


;
; Create a "Buffered Read" object to read the
; data, passing the metadata object handle
; created by vdf_create() as an argument
;
```

```
dfd = vdc_bufreadcreate(mfd)

;
; Determine the dimensions of the volume at
; the given transformation level.
;
; Note. vdc_getdim() correctly handles dimension
; calucation for volumes with non-power-of-two
; dimensions.
;
dim = vdc_getdim(dfd, reflevel)

; Create an appropriately sized array to hold the volume
;
f = fltarr(dim)
slice = fltarr(dim[0], dim[1])

; Prepare to read the indicated time step and variable
;
varnames = ['ml']
vdc_openvarread, dfd, 0, varnames[0], reflevel


; Read the volume one slice at a time
;
for z = 0, dim[2]-1 do begin
     vdc_bufreadslice, dfd, slice

     ; IDL won't let us read directly into a
     ; subscripted array - need
     ; to read into a 2D array and then copy to 3D :-(

     f[*,*,z] = slice
endfor

; Close the currently opened variable/time-step.
;
vdc_closevar, dfd

; Destroy the "buffered read" data transformation object.
; We're done with it.
;
vdc_bufreaddestroy, dfd
```

## *See Also*

VDC_BUFREADCREATE

# VDC_BUFWRITECREATE

This function creates a buffered write object for writing data to a VDC one slice at a time.

## *Syntax*

```
dfd = VDC_BUFWRITECREATE(mfd)
dfd = VDC_BUFWRITECREATE(vdffile)
```

## *Return Value*

The function returns a handle which may be used for subsequent buffered write operations.

## *Arguments*

### mfd

A metafile data handle (see the introduction).

### vdffile

The path to a VDC metafile (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDC_OPENVARWRITE, VDC_CLOSEVAR, VDC_BUFWRITESLICE, VDC_BUFWRITEDESTROY

# VDC_BUFWRITEDESTROY

This procedure destroys a buffered write object previously created with VDC_BUFWRITECREATE

## *Syntax*

```
VDC_BUFWRITEDESTROY, dfd
```

## *Arguments*

### dfd

A VDC data handle (see the introduction).

## *Keywords*

## *Examples*

## *See Also*

VDC_BUFWRITECREATE

# VDC_BUFWRITESLICE

This procedure writes a single slice (2D array) of floating point data to a VDC. The VDC handle argument, *dfd*, determines the variable, time-step, and refinement level for the write based on the currently opened variable (See VDC_OPENVARWRITE). VDC_BUFWRITESLICE must be called repeatedly until the entire variable volume is written.

## *Syntax*

```
VDC_BUFWRITESLICE, dfd, slice
```

## *Arguments*

### dfd

A VDC data handle (see the introduction).

### Slice

A 2D floating point array of dimension Nx by Ny, where Nx and Ny are the X and Y coordinate dimensions of the volume (in voxels) at the desired refinement level associated with the opened VDC handle, *dfd*.

## *Keywords*

## *Examples*

The example below shows how to write a data volume to a VDC using the VDC_BUFWRITESLICE procedure.

```
; Create a "buffered write" data transformation object.
; The data transformation object will permit us to
; write (transform) raw data into the data set.
; The metadata for the data volumes is
; obtained from the metadata file we created
; previously. I.e. 'vdffile' must contain the
; path to a previously created .vdf
; file. The 'vdc_bufwritecreate' returns a
; handle, 'dfd', for subsequent operations.
;
dfd = vdc_bufwritecreate(vdffile)


; Create a synthetic data volume
;
f = create_data(dim[0], dim[1], dim[2])

; Prepare the data set for writing. We need to
```

```
; identify the time step
; and the name of the variable that we wish to store
;
vdc_openvarwrite, dfd, 0, varnames[0], -1


; Write (transform) the volume to the data set one
; slice at a time
;
for z = 0, dim[2]-1 do begin
    vdc_bufwriteslice, dfd, f[*,*,z]
endfor

; Close the currently opened variable/time-step.
; We're done writing to it
;
vdc_closevar, dfd

; Destroy the "buffered write" data
; transformation object. We're done with it.
;
vdc_bufwritedestroy, dfd
```

### *See Also*

VDC_BUFWRITECREATE

# VDC_CLOSEVAR

This procedure closes the currently opened variable associated with the handle, dfd,

## *Syntax*

```
VDC_CLOSEVAR, dfd
```

## *Arguments*

### dfd

A VDC data handle (see the introduction).

## *Keywords*

## *Examples*

## *See Also*

VDC_OPENVARREAD, VDC_OPENVARWRITE

# VDC_GETDIM

This function returns the dimensions of a variable in the VDC associated with the VDC handle, dfd, at the indicated refinement level. If the refinement level is -1, the dimensions returned are those of the native volume

## *Syntax*

```
Result = VDC_GETDIM(dfd, reflevel)
```

## *Return Value*

The function returns a three element array containing the X, Y, and Z coordinate dimensions (in voxels), respectively.

## *Arguments*

### dfd

A VDC data handle (see the introduction).

### reflevel

Requested refinement level (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

# VDC_ISVALIDREGION

This boolean function returns true (1) if the indicated volume subregion coordinates are valid for the VDC associated with the handle, dfd, and false (0) if the region is invalid (outside of the bounds of the volume).

## *Syntax*

```
Result = VDC_ISVALIDREGION(dfd, min, max, reflevel)
```

## *Return Value*

The function returns true (1) if the specified subregion is contained within the volume. Otherwise it returns false (0).

## *Arguments*

### dfd

A VDC data handle (see the introduction).

### min, max

A three-element array specifying the minimum and maximum coordinates (in voxels), respectively, of the volume subregion. Coordinates are specified relative to the refinement level.

### reflevel

Requested refinement level (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDC_GETDIM

# VDC_MAPUSER2VOX

This function maps coordinates in user-defined space to voxel coordinates.

## *Syntax*

```
VDC_MAPUSER2VOX, dfd, reflevel, timestep, ucoord
```

## *Return Value*

A three-element, IDL long array is returned containing the voxel coordinates of the point closest to the user defined coordinates, *ucoord*. Coordinates are specified relative to the indicated refinement level and time step.

## *Arguments*

### dfd

A VDC data handle (see the introduction).

### reflevel

Requested refinement level (see the introduction)

### timestep

The time step of the requested mapping (see the introduction)

### ucoord

A three-element, floating-point array specifying the X, Y, and Z spatial coordinates of a point contained within a volume. The coordinates are specified relative to the user-defined coordinate system. The domain for the user-defined coordinate system may be retrieve via the VDF_GETEXTENTS function.

## *Keywords*

## *Examples*

## *See Also*

VDF_GETEXTENTS

# VDC_MAPVOX2USER

This function maps the integer coordinates of the specified voxel to floating point coordinates in user-defined space. Coordinates are specified relative to the indicated refinement level and time step.

## *Syntax*

```
VDC_MAPVOX2USER, dfd, reflevel, timestep, vcoord
```

## *Return Value*

A three-element, floating-point array is returned containing the coordinates in user-defined space of the voxel specified by *vcoord*. The domain for the user-defined coordinate system may be retrieve via the VDF_GETEXTENTS function.

## *Arguments*

### dfd

A VDC data handle (see the introduction).

### reflevel

Requested refinement level (see the introduction)

### timestep

The time step of the requested mapping (see the introduction)

### vcoord

A three-element, IDL Long array specifying the X, Y, and Z spatial coordinates of a point contained within a volume. The coordinates are specified relative to the voxel coordinate system.

## *Keywords*

## *Examples*

## *See Also*

VDF_GETEXTENTS

# VDC_OPENVARREAD

This procedure opens a volume within the VDC associated with the handle *dfd* for reading. The volume to be opened is identified by a combination of time step, variable name, and refinement level.

## *Syntax*

```
VDC_OPENVARREAD, dfd, timestep, varname, reflevel
```

## *Arguments*

### dfd

A VDC data handle (see the introduction).

### reflevel

Requested refinement level (see the introduction)

### timestep

The time step of the volume (see the introduction)

### varname

Name of the variable to open (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDC_CLOSEVAR, VDC_BUFREADCREATE, VDC_REGREADCREAT

# VDC_OPENVARWRITE

This procedure opens a volume within the VDC associated with the handle *dfd* for writing. The volume to be opened is identified by a combination of time step, variable name, and refinement level.

## *Syntax*

```
VDC_OPENVARWRITE, dfd, timestep, varname, reflevel
```

## *Arguments*

### dfd

A VDC data handle (see the introduction).

### reflevel

Requested refinement level (see the introduction)

### timestep

The time step of the volume (see the introduction)

### varname

Name of the variable to open (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDC_CLOSEVAR, VDC_BUFWRITECREATE, VDC_REGWRITECREAT

# VDC_REGREAD

This procedure reads a subregion from the currently opened volume associated with the VDC handle, *dfd*. The handle, *dfd*, must have been returned as the result of a call to VDC_REGREADCREAT. Furthermore, *dfd* must have been opened for reading with a prior call to VDC_OPENVARREAD.

## *Syntax*

```
VDC_REGREAD, dfd, min, max, volume
```

## *Arguments*

### dfd

A VDC data handle (see the introduction).

### min, max

A three-element array specifying the minimum and maximum coordinates (in voxels), respectively, of the volume subregion. Coordinates are specified relative to the refinement level.

### volume

A 3D floating point array with dimensions *max-min*+1

## *Keywords*

## *Examples*

```
; The refinement level. A value of -1 implies the
; finest (native) resolution
;
reflevel = -1



;
;   Create a "Region Read" object to read the data
;
dfd = vdc_regreadcreate('test.vdf')

; Determine the dimensions of the volume at
; the given transformation level.
;
; Note. vdc_getdim() correctly handles dimension
; calculation for volumes with non-power-of-two dimensions.
;
dim = vdc_getdim(dfd, reflevel)
```

```
;   Compute the coordinates for the desired subregion.
; In this case, the first octant will be read
min = [0,0,0]
max = (dim / 2) - 1

;   Create an appropriately sized array to hold the volume
;
f = fltarr(dim/2)

;   Prepare to read the indicated time step and variable
;
varnames = ['ml']
timestep = 0
vdc_openvarread, dfd, timestep, varnames[0], reflevel

; Read the volume subregion. Note, unlike the
; buffered read/write objects, the "Region Reader"
; object does not read a single slice at a
; time -- it slurps in the entire in single call.
;
vdc_regread, dfd, min, max, f

; Close the currently opened variable/time-step.
;
vdc_closevar, dfd


; Destroy the "buffered read" data transformation object.
; We're done with it.
;
vdc_regreaddestroy, dfd
```

## *See Also*

VDC_REGREADCREAT, VDC_OPENVARREAD

# VDC_REGREADCREATE

This function creates a region reader object for reading 3D data from a VDC.

## *Syntax*

```
dfd = VDC_REGREADCREATE(mfd)
dfd = VDC_REGREADCREATE(vdffile)
```

## *Return Value*

The function returns a handle which may be used for subsequent buffered write operations.

## *Arguments*

### mfd

A metafile data handle (see the introduction).

### vdffile

The path to a VDC metafile (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDC_OPENVARWRITE, VDC_CLOSEVAR, VDC_REGREAD, VDC_REGREADDESTROY

# VDC_REGREADCREATE2D

This function creates a region reader object for reading 2D data from a VDC.

## *Syntax*

```
dfd = VDC_REGREADCREATE(mfd)
dfd = VDC_REGREADCREATE(vdffile)
```

## *Return Value*

The function returns a handle which may be used for subsequent buffered write operations.

## *Arguments*

### mfd

A metafile data handle (see the introduction).

### vdffile

The path to a VDC metafile (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDC_OPENVARWRITE, VDC_CLOSEVAR, VDC_REGREAD, VDC_REGREADDESTROY

# VDC_REGREADDESTROY

This procedure destroys a region read object previously created with VDC_REGREADCREATE or VDC_REGREADCREATE2D

## *Syntax*

```
VDC_REGREADDESTROY, dfd
```

## *Arguments*

### dfd

A VDC data handle (see the introduction).

## *Keywords*

## *Examples*

## *See Also*

VDC_REGREADCREATE, VDC_REGREADCREATE2D

# VDC_REGWRITE

This procedure writes a subregion to the currently opened volume associated with the VDC handle, *dfd*. The handle, *dfd*, must have been returned as the result of a call to VDC_REGWRITECREAT. Furthermore, *dfd* must have been opened for writing with a prior call to VDC_OPENVARWRITE.

## *Syntax*

```
VDC_REGWRITE, dfd, volume, vcoord
```

## *Arguments*

### dfd

A VDC data handle (see the introduction).

### volume

A 3D floating point array containing the volume subregion to write to the VDC

### vcoord

A three-element, IDL Long array specifying the X, Y, and Z spatial coordinates of a point contained within a volume. The coordinates are specified relative to the voxel coordinate system. vcoord is used as the starting offset within the volume to write the subvolume.

## *Keywords*

## *Examples*

## *See Also*

VDC_REGREADCREAT, VDC_OPENVARREAD

# VDC_REGWRITECREATE

This function creates a region reader object for writing 3D data from a VDC.

## *Syntax*

```
dfd = VDC_REGREADCREATE(mfd)
dfd = VDC_REGREADCREATE(vdffile)
```

## *Return Value*

The function returns a handle, which may be used for subsequent buffered write operations.

## *Arguments*

### mfd

A metafile data handle (see the introduction).

### vdffile

The path to a VDC metafile (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDC_OPENVARWRITE, VDC_CLOSEVAR, VDC_REGWRITE, VDC_REGWRITEDESTROY

# VDC_REGWRITECREATE2D

This function creates a region reader object for writing 2D data from a VDC.

## *Syntax*

```
dfd = VDC_REGREADCREATE(mfd)
dfd = VDC_REGREADCREATE(vdffile)
```

## *Return Value*

The function returns a handle, which may be used for subsequent buffered write operations.

## *Arguments*

### mfd

A metafile data handle (see the introduction).

### vdffile

The path to a VDC metafile (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDC_OPENVARWRITE, VDC_CLOSEVAR, VDC_REGWRITE, VDC_REGWRITEDESTROY

# VDC_REGWRITEDESTROY

This procedure destroys a region write object previously created with VDC_REGWRITECREATE or VDC_REGWRITECREATE2D

## *Syntax*

```
VDC_REGWRITEDESTROY, dfd
```

## *Arguments*

### dfd

A VDC data handle (see the introduction).

## *Keywords*

## *Examples*

## *See Also*

VDC_REGWRITECREATE, VDC_REGWRITECREATE2D

# VDC_VAREXISTS

This Boolean function returns true (1) if the indicated volume exists (resides on disk) within the VDC associated with the VDC handle, *dfd*. False (0) is returned if the named volume is not present at the specified time step and refinement level.

## *Syntax*

```
VDC_VAREXISTS, dfd, timestep, varname, reflevel
```

## *Return Value*

Return true if the volume is present, false otherwise.

## *Arguments*

### dfd

A VDC data handle (see the introduction).

### reflevel

Requested refinement level (see the introduction)

### timestep

The time step of the requested mapping (see the introduction)

### varname

The name of the variable.

## *Keywords*

## *Examples*

## *See Also*

VDC_BUFREADCREATE, VDC_BUFWRITECREATE, VDC_REGREADCREATE, VDC_REGWRITECREATE

# VDF_CREATE

This function creates a metadata object for a VDC. The first form of the function creates a metadata object from an existing metadata file, *vdffile*. The second form creates a metadata object from scratch.

## *Syntax*

```
mfd = VDF_CREATE(vdffile)

mfd = VDF_CREATE(dim, reflevel, [BS=[bs]],
[NFILTERCOEF=nfiltercoef],
[NLIFTINGCOEF=nliftingcoef])
```

## *Return Value*

The function returns a handle which may be used for subsequent operations by VDF_ commands.

## *Arguments*

### vdffile

The path to a VDC metafile (see the introduction)

### dim

A three-element integer array specifying dimensions, in voxels, of all data volumes contained in the VDC.

### reflevel

The number of refinement levels for each data volume in the VDC. A value of zero would indicate a single resolution (the native resolution).

## *Keywords*

### BS

This keyword expects a three-element integer array specifying the blocking factor used for internal data storage. Presently the dimensions are required to be a power of two.

### NFILTERCOEF

The number of filter coefficients employed by wavelet transforms when creating approximations of the data in the VDC.

### NLIFTINGCOEF

The number of lifting coefficients employed by wavelet transforms when creating approximations of the data in the VDC.

### *Examples*

### *See Also*

VDF_DESTROY

# VDF_DESTROY

This function destroys a metadata object, *mfd*, previously created via a call to VDF_CREATE.

## *Syntax*

```
mfd = VDF_DESTROY, mfd
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE

# VDF_GETBLOCKSIZE

This function returns the internal blocking factor of the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETBLOCKSIZE(mfd)
```

## *Return Value*

The function returns three-element array containing the internal blocking factor.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

# VDF_GETCOMMENT

This function returns any user-defined global comment, if defined, for the VDC metadata object associated with the metadata handle, *mfd*. If not global comment is defined, the empty string is returned.

## *Syntax*

```
Result = VDF_GETCOMMENT(mfd)
```

## *Return Value*

The function returns a string.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETCOMMENT

# VDF_GETCOORDTYPE

This function returns the coordinate system type for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETCOORDTYPE(mfd)
```

## *Return Value*

The function returns string indicating the type of coordinate system.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETCOORDTYPE

# VDF_GETDBL

This function returns any double precision, global user-defined data, for the key, *tag*, for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETDBL(mfd, tag)
```

## *Return Value*

The function returns a double precision array of values.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### tag

A user defined key (see the introduction)

## *Keywords*

## *Examples*

```
;
; Print all global, user-defined double precision data
;
tags = vdf_getdbltags()
ntags = size(tags, /N_ELEMENTS)
for i = 0,ntags-1 do begin
     print, 'tag : ', tags[i]
     data = vdf_getdbl(tags[i])
     ndata = size(data, /N_ELEMENTS)
     for j=0,ndata-1 do begin
          print, data[j],
     endfor
endfor
```

## *See Also*

VDF_CREATE, VDF_SETDBL, VDF_GETDBLTAGS

# VDF_GETDBLTAGS

This function returns a list of global keys for double precision data for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETDBLTAGS(mfd)
```

## *Return Value*

The function returns an array of character strings.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETDBL, VDF_GETDBL

# VDF_GETDIMENSION

This function returns the X,Y,Z coordinate dimensions (in voxels) of all data volumes for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETDIMENSION(mfd)
```

## *Return Value*

The function returns three-element array.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE

# VDF_GETEXTENTS

This function returns the user extents for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETEXTENTS(mfd)
```

## *Return Value*

The function returns a six element, floating-point array containing the elements: [*MinX, MinY, MinZ, MaxX, MaxY, MaxZ*], the minimum and maximum bounds of the data volume expressed in the user defined coordinate system.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETEXTENTS

# VDF_GETFILTERCOEF

This function returns wavelet filter coefficients for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETFILTERCOEF(mfd)
```

## *Return Value*

The function returns a scalar value.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_GETLIFTINGCOEF

# VDF_GETGRIDTYPE

This function returns the grid type for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETGRIDTYPE(mfd)
```

## *Return Value*

The function returns string indicating the type of grid.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETGRIDTYPE

# VDF_GETLIFTINGCOEF

This function returns wavelet lifting coefficients for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETLIFTINGCOEF(mfd)
```

## *Return Value*

The function returns a scalar value.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_GETFILTERCOEF

# VDF_GETLONG

This function returns any Long global user-defined data, for the key, *tag*, for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETLONG(mfd, tag)
```

## *Return Value*

The function returns a Long array of values.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### tag

A user defined key (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETLONG, VDF_GETLONGTAGS

# VDF_GETLONGTAGS

This function returns a list of global keys for Long precision for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETLONGTAGS(mfd)
```

## *Return Value*

The function returns an array of character strings.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETLONG, VDF_GETLONG

# VDF_GETMAPPROJECTION

This function returns the 2D map projection string, if defined, for the VDC metadata object associated with the metadata handle, *mfd*. If not defined, the empty string is returned.

## *Syntax*

```
Result = VDF_GETMAPPROJECTION(mfd)
```

## *Return Value*

The function returns a string.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *Version History*

Introduce VAPOR 1.4.4

## *See Also*

VDF_CREATE, VDF_SETMAPPROJECTION

# VDF_GETNUMTIMESTEPS

This function returns the number of time steps for the VDC metadata object associated with the metadata handle, *mfd*. This value is an upper bound on the number of time steps the VDC may contain. The number of time steps present in the VDC at any time may be equal to or less than this number.

## *Syntax*

```
Result = VDF_GETNUMTIMESTEPS(mfd)
```

## *Return Value*

The function returns an integer result.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETEXTENTS

# VDF_GETNUMTRANSFORMS

This function returns the number of transforms (refinement levels) for the VDC metadata object associated with the metadata handle, *mfd*. A value of zero indicates no transforms (the data are stored at native resolution). A value of 1 indicates a single data coarsening, and so on.

## *Syntax*

```
Result = VDF_GETLIFTINGCOEF(mfd)
```

## *Return Value*

The function returns a scalar value.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE

# VDF_GETPERIODIC

This function returns a three-element array. Each element of the array is a Boolean indicating whether the data are periodic along the X, Y, and Z coordinate axes (YZ, XZ, and XY planes), respectively. A value of zero indicates non-periodic data. A non-zero value indicates periodic boundaries.

## *Syntax*

```
Result = VDF_PERIODIC(mfd)
```

## *Return Value*

The function returns a scalar value.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *Version History*

Introduce VAPOR 1.1

## *See Also*

VDF_CREATE, VDF_SETPERIODIC

# VDF_GETSTR

This function returns any character string, global user-defined data, for the key, *tag*, for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETSTR(mfd, tag)
```

## *Return Value*

The function returns a string

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### tag

A user defined key (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETSTR, VDF_GETSTRTAGS

# VDF_GETSTRTAGS

This function returns a list of global keys for string data for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETSTRTAGS(mfd)
```

## *Return Value*

The function returns an array of character strings.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETSTR, VDF_GETSTR

# VDF_GETTCOMMENT

This function returns the comment, for a time step, if it exists, for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETTCOMMENT(mfd, timestep)
```

## *Return Value*

These functions return a character string.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### timestep

The time step of the volume (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETTCOMMENT

# VDF_GETTDBL

This function returns any double precision, time step dependent, user-defined data, for the time step, *timestep*, and the key, *tag*, for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETTDBL(mfd, timestep, tag)
```

## *Return Value*

The function returns a double precision array of values.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### timestep

The time step of the volume (see the introduction)

### tag

A user defined key (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETTDBL, VDF_GETTDBLTAGS

# VDF_GETTDBLTAGS

This function returns a list of time step dependent keys for double precision data for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETTDBLTAGS(mfd)
```

## *Return Value*

The function returns an array of character strings.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETTDBL, VDF_GETTDBL

# VDF_GETTEXTENTS

This function returns the time-dependent, user extents (if defined) for the VDC metadata object associated with the metadata handle, *mfd*, at time step, *timestep*.

## *Syntax*

```
Result = VDF_GETTEXTENTS(mfd, timestep)
```

## *Return Value*

If time-dependent user extents are defined for the specified time step this function returns a six element, floating-point array containing the elements: [*MinX, MinY, MinZ, MaxX, MaxY, MaxZ*], the minimum and maximum bounds of the data volume expressed in the user defined coordinate system.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### timestep

The time step of the volume (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETTEXTENTS

# VDF_GETTLONG

This function returns any Long, time step dependent, user-defined data, for the time step, *timestep*, and the key, *tag*, for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETTLONG(mfd, timestep, tag)
```

## *Return Value*

The function returns an array of Longs.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### timestep

The time step of the volume (see the introduction)

### tag

A user defined key (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETTLONG, VDF_GETTLONGTAGS

# VDF_GETTLONGTAGS

This function returns a list of time step dependent keys for Long data for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETTLONGTAGS(mfd)
```

## *Return Value*

The function returns an array of character strings.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETTLONG, VDF_GETTLONG

# VDF_GETTSTR

This function returns any character string, time step dependent, user-defined data, for the time step, *timestep*, and the key, *tag*, for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETTSTR(mfd, timestep, tag)
```

## *Return Value*

The function returns a character string.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### timestep

The time step of the volume (see the introduction)

### tag

A user defined key (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETTSTR, VDF_GETTSTRTAGS

# VDF_GETTSTRTAGS

This function returns a list of time step dependent keys for character string data for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETTDBLTAGS(mfd)
```

## *Return Value*

The function returns an array of character strings.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETTSTR, VDF_GETTSTR

# VDF_GETTUSERTIME

This function returns the time of a time step for the VDC metadata object associated with the metadata handle, *mfd*. The time is a floating point value that correlates a time step, an integer offset, to a user-defined time value.

## *Syntax*

```
Result = VDF_GETTSUERTIME(mfd, timestep)
```

## *Return Value*

The function returns a floating point value.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### timestep

The time step of the volume (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETTUSERTIME

# VDF_GETT{XYZ}COORDS

These functions return the X,Y, and Z coordinates arrays, of a time step, if they exist for the VDC metadata object associated with the metadata handle, *mfd*. The coordinate arrays specify the coordinates of each voxel in the user-defined coordinate system. The coordinate arrays are only present if the grid type is *stretched*

## *Syntax*

```
Result = VDF_GETTXCOORDS(mfd, timestep)
Result = VDF_GETTYCOORDS(mfd, timestep)
Result = VDF_GETTZCOORDS(mfd, timestep)
```

## *Return Value*

These functions return arrays of floating point values.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### timestep

The time step of the volume (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETT{X,Y,Z}COORDS, VDF_GETGRIDTYPE

# VDF_GETVARIABLES2DXY

This function returns the names of the 2DXY variables for the VDC metadata object associated with the metadata handle, *mfd*. The list returned are the names of all of the 2DXY variables that the VDC may contain. The variables that are actually present in a VDC are a subset of this list.

## *Syntax*

```
Result = VDF_GETVARIABLES2DXY(mfd)
```

## *Return Value*

The function returns an array of strings.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *Version History*

1.4: introduced

## *See Also*

VDF_CREATE, VDF_SETVARNAMES, VDF_GETVARIABLES3D, VDF_GETVARIABLES2DXY, VDF_GETVARIABLES2DXZ, VDF_GETVARIABLES2DYZ, VDF_SETVARIABLES3D, VDF_SETVARIABLES2DXY, VDF_SETVARIABLES2DXZ, VDF_SETVARIABLES2DYZ

# VDF_GETVARIABLES2DXZ

This function returns the names of the 2DXZ variables for the VDC metadata object associated with the metadata handle, *mfd*. The list returned are the names of all of the 2DXZ variables that the VDC may contain. The variables that are actually present in a VDC are a subset of this list.

## *Syntax*

```
Result = VDF_GETVARIABLES2DXZ(mfd)
```

## *Return Value*

The function returns an array of strings.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *Version History*

1.4: introduced

## *See Also*

VDF_CREATE, VDF_SETVARNAMES, VDF_GETVARIABLES3D, VDF_GETVARIABLES2DXY, VDF_GETVARIABLES2DXZ, VDF_GETVARIABLES2DYZ, VDF_SETVARIABLES3D, VDF_SETVARIABLES2DXY, VDF_SETVARIABLES2DXZ, VDF_SETVARIABLES2DYZ

# VDF_GETVARIABLES2DYZ

This function returns the names of the 2DYZ variables for the VDC metadata object associated with the metadata handle, *mfd*. The list returned are the names of all of the 2DYZ variables that the VDC may contain. The variables that are actually present in a VDC are a subset of this list.

## *Syntax*

```
Result = VDF_GETVARIABLES2DYZ(mfd)
```

## *Return Value*

The function returns an array of strings.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *Version History*

1.4: introduced

## *See Also*

VDF_CREATE, VDF_SETVARNAMES, VDF_GETVARIABLES3D, VDF_GETVARIABLES2DXY, VDF_GETVARIABLES2DXZ, VDF_GETVARIABLES2DYZ, VDF_SETVARIABLES3D, VDF_SETVARIABLES2DXY, VDF_SETVARIABLES2DXZ, VDF_SETVARIABLES2DYZ

# VDF_GETVARIABLES3D

This function returns the names of the 3D variables for the VDC metadata object associated with the metadata handle, *mfd*. The list returned are the names of all of the 3D variables that the VDC may contain. The variables that are actually present in a VDC are a subset of this list.

## *Syntax*

```
Result = VDF_GETVARIABLES3D(mfd)
```

## *Return Value*

The function returns an array of strings.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *Version History*

1.4: introduced

## *See Also*

VDF_CREATE, VDF_SETVARNAMES, VDF_GETVARIABLES3D, VDF_GETVARIABLES2DXY, VDF_GETVARIABLES2DXZ, VDF_GETVARIABLES2DYZ, VDF_SETVARIABLES3D, VDF_SETVARIABLES2DXY, VDF_SETVARIABLES2DXZ, VDF_SETVARIABLES2DYZ

# VDF_GETVARNAMES

This function returns the names of the variables for the VDC metadata object associated with the metadata handle, *mfd*. The list returned are the names of all of the 2D and 3D variables that the VDC may contain. The variables that are actually present in a VDC are a subset of this list.

## *Syntax*

```
Result = VDF_GETVARNAMES(mfd)
```

## *Return Value*

The function returns an array of strings.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETVARNAMES, VDF_GETVARIABLES3D, VDF_GETVARIABLES2DXY, VDF_GETVARIABLES2DXZ, VDF_GETVARIABLES2DYZ, VDF_SETVARIABLES3D, VDF_SETVARIABLES2DXY, VDF_SETVARIABLES2DXZ, VDF_SETVARIABLES2DYZ

# VDF_GETVCOMMENT

This function returns the comment, for a variable, at a given time step, if it exists, for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETTCOMMENT(mfd, timestep, varname)
```

## *Return Value*

These functions return a character string.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### timestep

The time step of the volume (see the introduction)

### varname

A character string identifying the name of a variable within a VDC.

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETVCOMMENT

# VDF_GETVDBL

This function returns any double precision, time step dependent, user-defined data, for the variable, *varname*, at the time step, *timestep*, and the key, *tag*, for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETVDBL(mfd, timestep, varname, tag)
```

## *Return Value*

The function returns a double precision array of values.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### timestep

The time step of the volume (see the introduction)

### tag

A user defined key (see the introduction)

### varname

A character string identifying the name of a variable within a VDC.

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETVDBL, VDF_GETVDBLTAGS

# VDF_GETVDBLTAGS

This function returns a list of variable name dependent keys for double precision data for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETVDBLTAGS(mfd)
```

## *Return Value*

The function returns an array of character strings.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETVDBL, VDF_GETVDBL

# VDF_GETVLONG

This function returns any Long, time step dependent, user-defined data, for the variable, *varname*, at the time step, *timestep*, and the key, *tag*, for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETVLONG(mfd, timestep, varname, tag)
```

## *Return Value*

The function returns an array of Longs.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### timestep

The time step of the volume (see the introduction)

### tag

A user defined key (see the introduction)

### varname

A character string identifying the name of a variable within a VDC.

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETVLONG, VDF_GETVLONGTAGS

# VDF_GETVLONGTAGS

This function returns a list of variable name dependent keys for Long data for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETVLONGTAGS(mfd)
```

## *Return Value*

The function returns an array of character strings.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETVLONG, VDF_GETVLONG

# VDF_GETVSTR

This function returns any character string, time step dependent, user-defined data, for the variable, *varname*, at the time step, *timestep*, and the key, *tag*, for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETVSTR(mfd, timestep, varname, tag)
```

## *Return Value*

The function returns a character string.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### timestep

The time step of the volume (see the introduction)

### tag

A user defined key (see the introduction)

### varname

A character string identifying the name of a variable within a VDC.

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETVSTR, VDF_GETVSTRTAGS

# VDF_GETVSTRTAGS

This function returns a list of variable name dependent keys for character string data for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
Result = VDF_GETVSTRTAGS(mfd)
```

## *Return Value*

The function returns an array of character strings.

## *Arguments*

### mfd

A metadata object handle (see the introduction)

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_SETVSTR, VDF_GETVSTR

# VDF_SETCOMMENT

This procedure permits the specification of a global level comment for the VDC metadata object associated with the metadata handle, *mfd*. The comment is simply a character string and may contain any data the user wants. The data is not interpreted in any way by the VDC.

## *Syntax*

```
VDF_SETCOMMENT, mfd, comment
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### comment

A character string

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_GETCOMMENT

# VDF_SETCOORDTYPE

This procedure permits the specification of the coordinate system type for the VDC metadata object associated with the metadata handle, *mfd*. The coordinate system type, one of **Cartesian** or **Spherical**, classifies the geometry of data grids contained in the VDC. The default value is **Cartesian**.

## *Syntax*

```
VDF_SETCOORDTYPE, mfd, coordtype
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### coordtype

A character string specifying the coordinate system type of the VDC. Valid values are **cartesian** or **spherical**.

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_GETCOORDTYPE

# VDF_SETDBL

This procedure permits the specification of global-level double precision metadata for the VDC metadata object associated with the metadata handle, *mfd*. These metadata are not subject to interpretation by the VDC.

## *Syntax*

```
VDF_SETDBL, mfd, tag, data
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### tag

A user defined key (see the introduction)

### data

An array of double precision data.

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_GETDBL

# VDF_SETEXTENTS

This procedure sets the spatial domain extents in user-defined (world) coordinates for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
VDF_SETEXTENTS, mfd, extents
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### extents

A six-element array, the first three elements specify the minimum coordinate extents, the last three elements specify the maximum coordinate extents.

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_GETEXTENTS

# VDF_SETGRIDTYPE

This procedure permits the specification of the grid type for the VDC metadata object associated with the metadata handle, *mfd*. The grid type, one of **regular**, **spherical**, or **block_amr**, classifies the geometry of data grids contained in the VDC. The default value is **regular**.

## *Syntax*

```
VDF_SETGRIDTYPE, mfd, gridtype
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### gridtype

A character string specifying the grid type of the VDC. Valid values are **regular**, **stretched,** or **block_amr**. A regular grid is one where the spacing between samples is uniform and is implicitly defined by the resolution (number of grid points along each coordinate axes) and the spatial domain extents. A stretched grid is one where each X,Y, Z grid point coordinate is given by a monotonically increasing function of the grid point's integer offset. The coordinates are expressed in user-defined world coordinates. Lastly, a **block_amr** grid is one employing block structured Adaptive Mesh Refinement.

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_GETGRIDTYPE

# VDF_SETLONG

This procedure permits the specification of global-level Long precision integer metadata for the VDC metadata object associated with the metadata handle, *mfd*. These metadata are not subject to interpretation by the VDC.

## *Syntax*

```
VDF_SETLONG, mfd, tag, data
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### tag

A user defined key (see the introduction)

### data

An array of Long precision integer data.

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_GETLONG

# VDF_SETMAPPROJECTION

This procedure defines a 2D map projection from spherical to 2D space. The projection string defines a mapping from a portion of the earth's surface to a plane

## *Syntax*

```
VDF_SETMAPPROJECTION, mfd, projection_args
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### projection_args

The format of the string is white-space separated list of parameter names and values. I.e.

```
+param_name=param_value [+param_name=param_value...],
```

where *param_name* is the name of the parameter and *param_value* is the value associated with *param_name*. Further documentation on allowable parameters and values may be found in the references below.

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_GETMAPPROJECTION,
http://trac.osgeo.org/geotiff/
http://www.vapor.ucar.edu

# VDF_SETNUMTIMESTEPS

This procedure sets the maximum number of time steps for the VDC metadata object associated with the metadata handle, *mfd*. This value is an upper-bound on the number of time steps a VDC may contain. Valid VDCs may contain less than this maximum.

## *Syntax*

```
VDF_SETNUMTIMESTEPS, mfd, nmax
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### nmax

Maximum number of time steps.

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_GETNUMTIMESTEPS

# VDF_SETPERIODIC

This procedure the periodicity of the grid boundaries for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
VDF_SETPERIODIC, mfd, periodic
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### periodic

A three-element Boolean vector whose elements indicate whether the data are periodic along the X, Y, and Z coordinate axes, respectively.

## *Keywords*

## *Examples*

## *Version History*

Introduced VAPOR 1.1

## *See Also*

VDF_CREATE, VDF_GETPERIODIC

# VDF_SETSTR

This procedure permits the specification of global-level character string metadata for the VDC metadata object associated with the metadata handle, *mfd*. These metadata are not subject to interpretation by the VDC.

## *Syntax*

```
VDF_SETSTR, mfd, tag, data
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### tag

A user defined key (see the introduction)

### data

A character string.

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_GETLONG

# VDF_SETTCOMMENT

This procedure permits the specification of a time-step dependent comment for the VDC metadata object associated with the metadata handle, *mfd*. The comment is simply a character string and may contain any data the user wants. The data is not interpreted in any way by the VDC.

## *Syntax*

```
VDF_SETTCOMMENT, mfd, timestep, comment
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### timestep

The time step of the volume (see the introduction)

### comment

A character string.

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_GETTCOMMENT

# VDF_SETTDBL

This procedure permits the specification of time-step dependent, double precision metadata for the VDC metadata object associated with the metadata handle, *mfd*. These metadata are not subject to interpretation by the VDC.

## *Syntax*

```
VDF_SETDBL, mfd, tag, timestep, data
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### tag

A user defined key (see the introduction)

### timestep

The time step of the volume (see the introduction)

### data

An array of double precision data.

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_GETTDBL

# VDF_SETTEXTENTS

This procedure permits the specification of time-dependent spatial domain extents in user-defined (world) coordinates for the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
VDF_SETTEXTENTS, mfd, timestep, extents
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### timestep

The time step of the volume (see the introduction)

### extents

A six-element array, the first three elements specify the minimum coordinate extents, the last three elements specify the maximum coordinate extents.

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_GETTEXTENTS

# VDF_SETTLONG

This procedure permits the specification of time-step dependent, Long precision integer metadata for the VDC metadata object associated with the metadata handle, *mfd*. These metadata are not subject to interpretation by the VDC.

## *Syntax*

```
VDF_SETLONG, mfd, timestep, tag data
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### timestep

The time step of the volume (see the introduction)

### tag

A user defined key (see the introduction)

### data

An array of long precision integer data.

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_GETTLONG

# VDF_SETTSTR

This procedure permits the specification of time-step dependent, character string metadata for the VDC metadata object associated with the metadata handle, *mfd*. These metadata are not subject to interpretation by the VDC.

## *Syntax*

```
VDF_SETTSTR, mfd, timestep, tag, data
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### timestep

The time step of the volume (see the introduction)

### tag

A user defined key (see the introduction)

### data

A character string.

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_GETTSTR

# VDF_SETTUSERTIME

This procedure sets the user time (world coordinates) for a time step in the VDC metadata object associated with the metadata handle, *mfd*.

## *Syntax*

```
VDF_SETTUSERTIME, mfd, timestep, time
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### timestep

The time step of the volume (see the introduction)

### time

The time, in user-defined coordinates, of this time step. The user time may be any value. However, subsequent processing of the VDC may yield unexpected results if successive time steps to not have monotonically increasing or decreasing user times.

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_GETTUSERTIME

# VDF_SETT{X,Y,Z}COORDS

These procedures define the X, Y, and Z dimension coordinates for a stretched grid in the VDC metadata object associated with the metadata handle, *mfd*. These arrays are ignored if the grid type is not **stretched**.

## *Syntax*

```
VDF_SETTXCOORDS, mfd, timestep, coords
VDF_SETTYCOORDS, mfd, timestep, coords
VDF_SETTZCOORDS, mfd, timestep, coords
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### timestep

The time step of the volume (see the introduction)

### coords

An array of coordinate values for the associated coordinate dimension. Subsequent processing of the VDC may produce unexpected results if the values contained in the array are not monotonically increasing or decreasing.

## *Keywords*

## *Examples*

```
;
; Set the coordinate arrays for each time step to a monotonically
; increasing, uniformly spaced value.
;
for i=0,VDF_GETNUMTIMESTEPS(mfd)-1 do begin
     dim = VDF_GETDIMENSION(mfd)
     coord = findgen(dim[0])
     VDF_SETTXCOORDS,mfd, timestep, coord
     coord = findgen(dim[1])
     VDF_SETTYCOORDS,mfd, timestep, coord
     coord = findgen(dim[2])
     VDF_SETTZCOORDS,mfd, timestep, coord
endfor
```

## *See Also*

VDF_CREATE, VDF_GETT{XYZ}COORDS, VDF_SETGRIDTYPE

# VDF_SETVARIABLES2DXY

This procedure indicates which variables are of type 2DXY in the VDC metadata object associated with the metadata handle, *mfd*. The variable names must have been previously set with VDF_SETVARNAMES.

## *Syntax*

```
VDF_SETVARIABLES2DXY, mfd, varnames
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### varnames

An array of character strings specifying which variables are of type 2DXY.

## *Keywords*

## *Examples*

```
;
; Establish the names and types of the variables in the VDF.
;
vars2d = ['height', 'ground_temp']
vars3d = ['vx', 'vy', 'vz']
vdf_setvarnames, mfd, [vars2d, vars3d]
vdf_setvariables3d, mfd, vars3d
vdf_setvariables2dxy, mfd, vars2d
```

## *Version History*

1.4: introduced

## *See Also*

VDF_CREATE, VDF_GETVARNAMES, VDF_GETVARIABLES3D, VDF_GETVARIABLES2DXY, VDF_GETVARIABLES2DXZ, VDF_GETVARIABLES2DYZ, VDF_SETVARIABLES3D, VDF_SETVARIABLES2DXY, VDF_SETVARIABLES2DXZ, VDF_SETVARIABLES2DYZ

# VDF_SETVARIABLES2DXZ

This procedure indicates which variables are of type 2DXZ in the VDC metadata object associated with the metadata handle, *mfd*. The variable names must have been previously set with VDF_SETVARNAMES.

## *Syntax*

```
VDF_SETVARIABLES2DXZ, mfd, varnames
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### varnames

An array of character strings specifying which variables are of type 2DXZ.

## *Keywords*

## *Examples*

## *Version History*

1.4: introduced

## *See Also*

VDF_CREATE, VDF_GETVARNAMES, VDF_GETVARIABLES3D, VDF_GETVARIABLES2DXY, VDF_GETVARIABLES2DXZ, VDF_GETVARIABLES2DYZ, VDF_SETVARIABLES3D, VDF_SETVARIABLES2DXY, VDF_SETVARIABLES2DXZ, VDF_SETVARIABLES2DYZ

# VDF_SETVARIABLES2DYZ

This procedure indicates which variables are of type 2DYZ in the VDC metadata object associated with the metadata handle, *mfd*. The variable names must have been previously set with VDF_SETVARNAMES.

## *Syntax*

```
VDF_SETVARIABLES2DYZ, mfd, varnames
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### varnames

An array of character strings specifying which variables are of type 2DYZ.

## *Keywords*

## *Examples*

## *Version History*

1.4: introduced

## *See Also*

VDF_CREATE, VDF_GETVARNAMES, VDF_GETVARIABLES3D, VDF_GETVARIABLES2DXY, VDF_GETVARIABLES2DXZ, VDF_GETVARIABLES2DYZ, VDF_SETVARIABLES3D, VDF_SETVARIABLES2DXY, VDF_SETVARIABLES2DXZ, VDF_SETVARIABLES2DYZ

# VDF_SETVARIABLES3D

This procedure indicates which variables are of type 3D in the VDC metadata object associated with the metadata handle, *mfd*. The variable names must have been previously set with VDF_SETVARNAMES. This procedure is largely superfluous as the default type of a variable is 3D. The procedure is included for completeness.

## *Syntax*

```
VDF_SETVARIABLES3D, mfd, varnames
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### varnames

An array of character strings specifying which variables are of type 3D.

## *Keywords*

## *Examples*

## *Version History*

1.4: introduced

## *See Also*

VDF_CREATE, VDF_GETVARNAMES, VDF_GETVARIABLES3D, VDF_GETVARIABLES2DXY, VDF_GETVARIABLES2DXZ, VDF_GETVARIABLES2DYZ, VDF_SETVARIABLES3D, VDF_SETVARIABLES2DXY, VDF_SETVARIABLES2DXZ, VDF_SETVARIABLES2DYZ

# VDF_SETVARNAMES

This procedure sets the name and the default type of the field variables in the VDC metadata object associated with the metadata handle, *mfd*. The default variable type is 3D.

## *Syntax*

```
VDF_SETVARNAMES, mfd, varnames
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### varnames

An array of character strings specifying the name, and indirectly the number, of each field variable in a VDC.

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_GETVARNAMES, VDF_GETVARIABLES3D, VDF_GETVARIABLES2DXY, VDF_GETVARIABLES2DXZ, VDF_GETVARIABLES2DYZ, VDF_SETVARIABLES3D, VDF_SETVARIABLES2DXY, VDF_SETVARIABLES2DXZ, VDF_SETVARIABLES2DYZ

# VDF_SETVCOMMENT

This procedure permits the specification of a time-step and variable dependent comment for the VDC metadata object associated with the metadata handle, *mfd*. The comment is simply a character string and may contain any data the user wants. The data is not interpreted in any way by the VDC.

## *Syntax*

```
VDF_SETVCOMMENT, mfd, timestep, varname, comment
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### timestep

The time step of the volume (see the introduction)

### varname

A character string identifying the name of a variable within a VDC.

### comment

A character string.

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_GETTCOMMENT

# VDF_SETVDBL

This procedure permits the specification of time-step and variable dependent, double precision metadata for the VDC metadata object associated with the metadata handle, *mfd*. These metadata are not subject to interpretation by the VDC.

## *Syntax*

```
VDF_SETVDBL, mfd, timestep, varname, tag, data
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### timestep

The time step of the volume (see the introduction)

### varname

A character string identifying the name of a variable within a VDC.

### tag

A user defined key (see the introduction)

### data

An array of double precision data.

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_GETVDBL

# VDF_SETVLONG

This procedure permits the specification of time-step and variable dependent, long precision integer metadata for the VDC metadata object associated with the metadata handle, *mfd*. These metadata are not subject to interpretation by the VDC.

## *Syntax*

```
VDF_SETVLONG, mfd, timestep, varname, tag, data
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### timestep

The time step of the volume (see the introduction)

### varname

A character string identifying the name of a variable within a VDC.

### tag

A user defined key (see the introduction)

### data

An array of long precision integer data.

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_GETVLONG

# VDF_SETVSTR

This procedure permits the specification of time-step and variable dependent, character string metadata for the VDC metadata object associated with the metadata handle, *mfd*. These metadata are not subject to interpretation by the VDC.

## *Syntax*

```
VDF_SETVSTR, mfd, timestep, varname, tag, data
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### timestep

The time step of the volume (see the introduction)

### varname

A character string identifying the name of a variable within a VDC.

### tag

A user defined key (see the introduction)

### data

A character string.

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE, VDF_GETVSTR

# VDF_WRITE

This procedure writes a VDF file for the VDC metadata object associated with the metadata handle, *mfd*. These metadata are not subject to interpretation by the VDC.

## *Syntax*

```
VDF_SETVSTR, mfd, vdffile
```

## *Arguments*

### mfd

A metadata object handle (see the introduction)

### vdffile

A character string specifying the path to the output VDF file. By convention, the file should have a .**vdf** extension.

## *Keywords*

## *Examples*

## *See Also*

VDF_CREATE