Reference Manual

Generated by Doxygen 1.6.1

Wed Feb 5 13:35:12 2014

Contents

Chapter 1

Todo List

Member LoadArrayData Currently only supports data mode follows

Member VTK_Write abstract to sub functions

Member VTK_WritePolydata assert that points are floats

Member vtkData_struct::scalar_data for now only support scalars and vectors

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 array_struct Struct Reference

Public Attributes

- unsigned char **type**
- unsigned char **category**
- int **rank**
- int **shape** [DX_MAX_RANK]
- int **items**
- unsigned char **endian**
- unsigned char **dataType**
- unsigned char **dataMode**
- char **file** [DX_MAX_TOKEN_LENGTH]
- int **offset**
- void * **data**

The documentation for this struct was generated from the following file:

- dxFileReader.h

4.2 attribute_struct Struct Reference

Public Attributes

- char **attribute_name** [DX_MAX_TOKEN_LENGTH]
- char **string** [DX_MAX_TOKEN_LENGTH]
- int **number**

The documentation for this struct was generated from the following file:

- dxFileReader.h

4.3 dxFile_struct Struct Reference

Public Attributes

- char * **filename**
- FILE * **fp**
- int **numObjects**
- object * **objs**

The documentation for this struct was generated from the following file:

- dxFileReader.h

4.4 field_struct Struct Reference

Public Attributes

- int **numComponents**
- object ** **components**

The documentation for this struct was generated from the following file:

- dxFileReader.h

4.5 gridconnections_struct Struct Reference

Public Attributes

- int **numCounts**
- int * **counts**

The documentation for this struct was generated from the following file:

- dxFileReader.h

4.6 gridpositions_struct Struct Reference

Public Attributes

- int **numCounts**
- int * **counts**
- float * **origin**
- float * **deltas**

The documentation for this struct was generated from the following file:

- dxFileReader.h

4.7 group_struct Struct Reference

Public Attributes

- int **numMembers**
- object ∗∗ **members**

The documentation for this struct was generated from the following file:

- dxFileReader.h

4.8 object_struct Struct Reference

Public Attributes

- unsigned char **class**
- char **alias** [DX_MAX_TOKEN_LENGTH]
- char **name** [DX_MAX_TOKEN_LENGTH]
- int **number**
- unsigned char **isLoaded**
- void * **obj**
- int **numAttributes**
- attribute * **attributes**
- fpos_t **pos**

The documentation for this struct was generated from the following file:

- dxFileReader.h

4.9 polydata_struct Struct Reference

Public Attributes

- int **numPoints**
- float * **points**
- int **numPolygons**
- int * **numVerts**
- int * **polygons**

The documentation for this struct was generated from the following file:

- vtkFileWriter.h

4.10 rectilinearGrid_struct Struct Reference

Public Attributes

- int **dimensions** [VTK_DIM]
- int **numX**
- int **numY**
- int **numZ**
- float * **X_coordinates**
- float * **Y_coordinates**
- float * **Z_coordinates**

The documentation for this struct was generated from the following file:

- vtkFileWriter.h

4.11 scalar_struct Struct Reference

Public Attributes

- char **name** [32]
- int **type**
- void * **data**

The documentation for this struct was generated from the following file:

- vtkFileWriter.h

4.12 series_struct Struct Reference

Public Attributes

- int **numMembers**
- float * **positions**
- object ** **members**

The documentation for this struct was generated from the following file:

- dxFileReader.h

4.13 structuredGrid_struct Struct Reference

Public Attributes

- int **dimensions** [VTK_DIM]
- int **numPoints**
- float * **points**

The documentation for this struct was generated from the following file:

- vtkFileWriter.h

4.14 structuredPoints_struct Struct Reference

Public Attributes

- int **dimensions** [VTK_DIM]
- float **origin** [VTK_DIM]
- float **spacing** [VTK_DIM]

The documentation for this struct was generated from the following file:

- vtkFileWriter.h

4.15 unstructuredGrid_struct Struct Reference

Public Attributes

- int **numPoints**
- float * **points**
- int **numCells**
- int **cellSize**
- int * **cells**
- int * **numVerts**
- int * **cellTypes**

The documentation for this struct was generated from the following file:

- vtkFileWriter.h

4.16 vector_struct Struct Reference

Public Attributes

- char **name** [32]
- int **type**
- void * **data**

The documentation for this struct was generated from the following file:

- vtkFileWriter.h

4.17 vtkData_struct Struct Reference

Public Attributes

- int **numScalars**
- int **numColorScalars**
- int **numLookupTables**
- int **numVectors**
- int **numNormals**
- int **numTextureCoords**
- int **numTensors**
- int **numFields**
- int **size**
- scalar * scalar_data
- vector * **vector_data**

4.17.1 Member Data Documentation

4.17.1.1 scalar* vtkData_struct::scalar_data

Todo

for now only support scalars and vectors

The documentation for this struct was generated from the following file:

- vtkFileWriter.h

4.18 vtkDataFile_struct Struct Reference

Public Attributes

- FILE * **fp**
- char **vtkVersion** [4]
- char **title** [VTK_TITLE_LENGTH]
- unsigned char **dataType**
- unsigned char **geometry**
- void * **dataset**
- vtkData * **pointdata**
- vtkData * **celldata**

The documentation for this struct was generated from the following file:

- vtkFileWriter.h

Chapter 5

File Documentation

5.1 dxFileReader.h File Reference

Reads an OpenDX data file. `#include <stdio.h>`

`#include <stdlib.h>`

`#include <string.h>`

`#include <endian.h>`

`#include <stdint.h>`

`#include "ioutils.h"`

Classes

- struct object_struct
- struct array_struct
- struct attribute_struct
- struct field_struct
- struct group_struct
- struct gridpositions_struct
- struct gridconnections_struct
- struct series_struct
- struct dxFile_struct

Defines

- #define **DX_MAX_FILENAME_LENGTH** 256
- #define **DX_MAX_TOKEN_LENGTH** 32
- #define **DX_MAX_MESH_DIMENSIONS** 6
- #define **DX_COMMENT_LENGTH** 256
- #define **DX_READ_BUFFER_SIZE** 2048
- #define **DX_SUCCESS** 1
- #define **DX_MEMORY_ERROR** 0
- #define **DX_FILE_NOT_FOUND_ERROR** -1
- #define **DX_INVALID_FILE_ERROR** -2

- #define **DX_NOT_SUPPORTED_ERROR** -3
- #define **DX_INVALID_USAGE_ERROR** -4
- #define **DX_NOT_IMPLEMENTED_YET_ERROR** -5
- #define **DX_FIELD** 0
- #define **DX_ATTRIBUTE** 1
- #define **DX_CONSTANTARRAY** 2
- #define **DX_ARRAY** 3
- #define **DX_REGULARARRAY** 4
- #define **DX_PRODUCTARRAY** 5
- #define **DX_GRIDPOSITIONS** 6
- #define **DX_PATHARRAY** 7
- #define **DX_MESHARRAY** 8
- #define **DX_GRIDCONNECTIONS** 9
- #define **DX_GROUP** 10
- #define **DX_SERIES** 11
- #define **DX_MAX_RANK** 10
- #define **DX_INT** 0
- #define **DX_FLOAT** 1
- #define **DX_INT_SIZE** 4
- #define **DX_FLOAT_SIZE** 4
- #define **DX_REAL** 0
- #define **DX_COMPLEX** 1
- #define **DX_LSB** 0
- #define **DX_MSB** 1
- #define **DX_TEXT** 0
- #define **DX_IEEE** 1
- #define **DX_BINARY** 2
- #define **DX_ASCII** 3
- #define **DX_OFFSET** 0
- #define **DX_FILE** 1
- #define **DX_FOLLOWS** 2
- #define **streq**(a, b) (strcmp((a),(b)) == 0)

Typedefs

- typedef struct object_struct **object**
- typedef struct attribute_struct **attribute**
- typedef struct field_struct **field**
- typedef struct array_struct **array**
- typedef struct group_struct **group**
- typedef struct gridpositions_struct **gridpositions**
- typedef struct gridconnections_struct **gridconnections**
- typedef struct series_struct **series**
- typedef struct dxFile_struct **dxFile**

Functions

- int DX_Open (dxFile *file, const char *filename)
  Opens an OpenDX file.
- int DX_LoadAll (dxFile *file)
  loads objects into memory
- int ParseObjectHeader (object *obj, char *header)
  Parses an object header.
- int ParseArrayObjectHeader (object *obj, char *header)
  Parses an array object header.
- int ParseFieldObjectHeader (object *obj, char *header)
  Parses a field object header.
- int ParseGroupObjectHeader (object *obj, char *header)
  Parses a group object header.
- int ParseGridPositionsObjectHeader (object *obj, char *header)
  Parses gridpositions object header.
- int ParseGridConnectionsObjectHeader (object *obj, char *header)
  Parses gridconnections object header.
- int ParseSeriesObjectHeader (object *obj, char *header)
  Parses series object header.
- int LoadObjectData (object *obj, dxFile *file)
  loads object data
- int LoadArrayData (object *obj, dxFile *file)
  loads array data
- int LoadFieldData (object *obj, dxFile *file)
  load field data
- int LoadGroupData (object *obj, dxFile *file)
  loads group data
- int LoadGridPositionsData (object *obj, dxFile *file)
  loads girdpositions data
- int LoadGridConnectionsData (object *obj, dxFile *file)
  loads array data
- int LoadSeriesData (object *obj, dxFile *file)
  loads series data

- void PrintObjectHeader (object *obj)

  prints the info from a loaded object header

- attribute * GetAttribute (object *obj, char *key)

  Gets an attribute if the object has such an attribute.

- object * GetObject (dxFile *file, char *name)

  Gets an object by name if it exists.

5.1.1 Detailed Description

Reads an OpenDX data file. This library implements a light weight reader for the native OpenDX data format. The complete standard is not necessarily implemented.

Author:

David J. Warne (david.warne@qut.edu.au)
High Performance Computing and Research Support
Queensland University of Technology

5.1.2 Function Documentation

5.1.2.1 int DX_LoadAll (dxFile * *file*)

loads objects into memory works through each object header and import data appropiately making links were appropriate.

Parameters:

file the file to load data from

Returns:

DX_SUCCESS on compeletion, otherwise an appropriate error message

5.1.2.2 int DX_Open (dxFile * *file*, const char * *filename*)

Opens an OpenDX file. reads through the file finding all objects and populating object descriptors. No data is actually loaded into memory.

Parameters:

filename The name of the OpenDX file

Returns:

DX_SUCCESS on compeletion, otherwise an appropriate error message

5.1.2.3 attribute* GetAttribute (object * *obj*, char * *key*)

Gets an attribute if the object has such an attribute.

Parameters:

obj the object to get the attribute of

key the name of the desired attribute

Returns:

a pointer to the attribute, NULL if the object does not have such an attribute

5.1.2.4 object* GetObject (dxFile * *file*, char * *name*)

Gets an object by name if it exists.

Parameters:

file the openDX file object

name the key name to search for

Returns:

a pointer to the object, NULL if the object does not exist

5.1.2.5 int LoadArrayData (object * *obj*, dxFile * *file*)

loads array data allocates memory and loads data array into memory

Parameters:

obj the pointer which wraps the array object

file the dxFile structure, assumes the stream cursor is located just after the array header.

Returns:

DX_SUCCESS on compeletion, otherwise an appropriate error message

Todo

Currently only supports data mode follows

5.1.2.6 int LoadFieldData (object * *obj*, dxFile * *file*)

load field data locates the components it the object array, assigns an alias to each and stores the pointer to associate the objects with the field.

Parameters:

obj the object pointer which wraps the field.

file the dxFile structure, assumes the stream cursor is located just after the field header.

Returns:

DX_SUCCESS on compeletion, otherwise an appropriate error message

Note:

This loader assumes all references are in the same dx file

5.1.2.7 int LoadGridConnectionsData (object * *obj*, dxFile * *file*)

loads array data allocates memory and loads data array into memory

Parameters:

obj the pointer which wraps the array object

file the dxFile structure, assumes the stream cursor is located just after the array header.

Returns:

DX_SUCCESS on compeletion, otherwise an appropriate error message

5.1.2.8 int LoadGridPositionsData (object * *obj*, dxFile * *file*)

loads girdpositions data allocates memory and loads gridpositions into memory

Parameters:

obj the pointer which wraps the array object

file the dxFile structure, assumes the stream cursor is located just after the array header.

Returns:

DX_SUCCESS on compeletion, otherwise an appropriate error message

5.1.2.9 int LoadGroupData (object * *obj*, dxFile * *file*)

loads group data locates the members it the object array, assigns an alias to each and stores the pointer to associate the objects with the group.

Parameters:

obj the object pointer which wraps the group.

file the dxFile structure, assumes the stream cursor is located just after the group header.

Returns:

DX_SUCCESS on compeletion, otherwise an appropriate error message

5.1.2.10 int LoadObjectData (object * *obj*, dxFile * *file*)

loads object data loads data via appropriate sub-function, then imports attributes and asserts isLoaded flag.

Parameters:

obj a pointer to the object to load

file the dxFile to load from

Returns:

DX_SUCCESS on compeletion, otherwise an appropriate error message

5.1.2.11 int LoadSeriesData (object * *obj*, dxFile * *file*)

loads series data allocates memory and loads series into memory

Parameters:

obj the pointer which wraps the array object

file the dxFile structure, assumes the stream cursor is located just after the array header.

Returns:

DX_SUCCESS on compeletion, otherwise an appropriate error message

5.1.2.12 int ParseArrayObjectHeader (object * *obj*, char * *header*)

Parses an array object header.

Parameters:

obj a pointer to the object wrapper that will hold this array

header the pointer to header line starting from type

Returns:

DX_SUCCESS if successfully complete, otherwise an appropriate error code

5.1.2.13 int ParseFieldObjectHeader (object * *obj*, char * *header*)

Parses a field object header.

Parameters:

obj a pointer to the object wrapper that will hold this field

header the pointer to header line starting from type

Returns:

DX_SUCCESS if successfully complete, otherwise an appropriate error code

5.1.2.14 int ParseGridConnectionsObjectHeader (object * *obj*, char * *header*)

Parses gridconnections object header.

Parameters:

obj a pointer to the object wrapper that will hold this gridconnections object

header the pointer to header line starting from type

Returns:

DX_SUCCESS if successfully complete, otherwise an appropriate error code

5.1.2.15 int ParseGridPositionsObjectHeader (object * *obj*, char * *header*)

Parses gridpositions object header.

Parameters:

obj a pointer to the object wrapper that will hold this gridpositions object

header the pointer to header line starting from type

Returns:

DX_SUCCESS if successfully complete, otherwise an appropriate error code

5.1.2.16 int ParseGroupObjectHeader (object * *obj*, char * *header*)

Parses a group object header.

Parameters:

obj a pointer to the object wrapper that will hold this group

header the pointer to header line starting from type

Returns:

DX_SUCCESS if successfully complete, otherwise an appropriate error code

5.1.2.17 int ParseObjectHeader (object * *obj*, char * *header*)

Parses an object header. This just tests what type of object it is and differs to an object specific parser. On successful execution, the object header information will be set.

Parameters:

obj a pointer to an object wrapper

header the header text line

Returns:

DX_SUCCESS or an appropiate error code

5.1.2.18 int ParseSeriesObjectHeader (object * *obj*, char * *header*)

Parses series object header.

Parameters:

obj a pointer to the object wrapper that will hold this series

header the pointer to header line starting from type

Returns:

DX_SUCCESS if successfully complete, otherwise an appropriate error code

5.1.2.19 void PrintObjectHeader (object * *obj*)

prints the info from a loaded object header this function is primarily for debugging purposes

Parameters:

obj a pointer to the object header wrapper

5.2 vtkFileWriter.h File Reference

Writes a vtk data file using the legacy format. `#include <stdio.h>`

`#include <stdlib.h>`

`#include <stdint.h>`

`#include <endian.h>`

Classes

- struct scalar_struct
- struct vector_struct
- struct vtkData_struct
- struct vtkDataFile_struct
- struct structuredPoints_struct
- struct structuredGrid_struct
- struct polydata_struct
- struct rectilinearGrid_struct
- struct unstructuredGrid_struct

Defines

- #define **VTK_ASCII** 0
- #define **VTK_BINARY** 1
- #define **H2BE32**(buf, size)
- #define **BE2H32**(buf, size)
- #define **VTK_STRUCTURED_POINTS** 0
- #define **VTK_STRUCTURED_GRID** 1
- #define **VTK_UNSTRUCTURED_GRID** 2
- #define **VTK_POLYDATA** 3
- #define **VTK_RECTILINEAR_GRID** 4
- #define **VTK_FIELD** 5
- #define **VTK_VERTEX** 1
- #define **VTK_POLY_VERTEX** 2
- #define **VTK_LINE** 3
- #define **VTK_POLY_LINE** 4
- #define **VTK_TRIANGLE** 5
- #define **VTK_TRIANGLE_STRIP** 6
- #define **VTK_POLYGON** 7
- #define **VTK_PIXEL** 8
- #define **VTK_QUAD** 9
- #define **VTK_TETRA** 10
- #define **VTK_VOXEL** 11
- #define **VTK_HEXAHEDRON** 12
- #define **VTK_WEDGE** 13
- #define **VTK_PYRAMID** 14
- #define **VTK_QUADRATIC_EDGE** 21
- #define **VTK_QUADRATIC_TRIANGLE** 22
- #define **VTK_QUADRATIC_QUAD** 23

- #define **VTK_QUADRATIC_TETRA** 24
- #define **VTK_QUADRATIC_HEXAHEDRON** 25
- #define **VTK_VERSION** "4.2"
- #define **VTK_TITLE_LENGTH** 256
- #define **VTK_DIM** 3
- #define **VTK_INT** 0
- #define **VTK_FLOAT** 1
- #define **VTK_SUCCESS** 1
- #define **VTK_MEMORY_ERROR** 0
- #define **VTK_FILE_NOT_FOUND_ERROR** -1
- #define **VTK_INVALID_FILE_ERROR** -2
- #define **VTK_NOT_SUPPORTED_ERROR** -3
- #define **VTK_INVALID_USAGE_ERROR** -4
- #define **VTK_NOT_IMPLEMENTED_YET_ERROR** -5
- #define **VTK_FILE_ERROR** -6
- #define **VTK_TYPE_DEFAULT** VTK_ASCII

Typedefs

- typedef struct vtkDataFile_struct **vtkDataFile**
- typedef struct unstructuredGrid_struct **unstructuredGrid**
- typedef struct structuredPoints_struct **structuredPoints**
- typedef struct polydata_struct **polydata**
- typedef struct vtkData_struct **vtkData**
- typedef struct scalar_struct **scalar**
- typedef struct vector_struct **vector**

Functions

- int VTK_Open (vtkDataFile *file, char *filename)
  Opens a vtk file for writing.
- int VTK_Write (vtkDataFile *file)
  exports the vtk data file object to a file
- int VTK_WriteUnstructuredGrid (FILE *fp, unstructuredGrid *ug, char type)
  writes a VTK unstructured grid to the file ouput stream
- int VTK_WritePolydata (FILE *fp, polydata *pd, char type)
  writes a VTK poly data mesh to the file output stream
- int VTK_WriteStructuredPoints (FILE *fp, structuredPoints *sp, char type)
  wriets a VTK structured point mesh
- int VTK_WriteData (FILE *fp, vtkData *data, char type)
  writes VTK data attributes (i.e., point or cell data)
- int VTK_Close (vtkDataFile *file)
  closes the vtk file

5.2.1 Detailed Description

Writes a vtk data file using the legacy format. This library implements a light weight legacy vtk format writer. This is intended to be used as a part of the dx2vtk program.

Author:

David J. Warne (david.warne@qut.edu.au)
High Performance Computing and Research Support
Queensland University of Technology

5.2.2 Define Documentation

5.2.2.1 #define BE2H32(buf, size)

Value:

```
{                                       \
    int iii;                            \
    uint32_t * buf32;                   \
    buf32 = (uint32_t *)(buf);          \
    for(iii=0;iii<(size);iii++)         \
    {                                   \
        buf32[iii] = be32toh(buf32[iii]); \
    }                                   \
}                                       \
```

5.2.2.2 #define H2BE32(buf, size)

Value:

```
{                                       \
    int iii;                            \
    uint32_t * buf32;                   \
    buf32 = (uint32_t *)(buf);          \
    for(iii=0;iii<(size);iii++)         \
    {                                   \
        buf32[iii] = htobe32(buf32[iii]); \
    }                                   \
}                                       \
```

5.2.3 Function Documentation

5.2.3.1 int VTK_Open (vtkDataFile * *file*, char * *filename*)

Opens a vtk file for writing.

Parameters:

file the vtk file object

filename the filename to open the object under

5.2.3.2 int VTK_Write (vtkDataFile * *file*)

exports the vtk data file object to a file

Todo

abstract to sub functions

5.2.3.3 int VTK_WriteData (FILE * *fp*, vtkData * *data*, char *type*)

writes VTK data attributes (i.e., point or cell data)

Parameters:

fp the ouptut file stream
data the vtkData struct
type specifies write mode, either VTK_ASCII or VTK_BINARY

5.2.3.4 int VTK_WritePolydata (FILE * *fp*, polydata * *pd*, char *type*)

writes a VTK poly data mesh to the file output stream

Parameters:

fp the file output stream
pd the polydata mesh
type specifies write mode, either VTK_ASCII or VTK_BINARY

Todo

assert that points are floats

5.2.3.5 int VTK_WriteStructuredPoints (FILE * *fp*, structuredPoints * *sp*, char *type*)

wriets a VTK structured point mesh

Parameters:

fp the output file stream
the structured point mesh
type specifies write mode, either VTK_ASCII or VTK_BINARY

5.2.3.6 int VTK_WriteUnstructuredGrid (FILE * *fp*, unstructuredGrid * *ug*, char *type*)

writes a VTK unstructured grid to the file ouput stream

Parameters:

fp the file output stream
ug the unstructured grid
type specifies write mode, either VTK_ASCII or VTK_BINARY

Index