A Description of the Classes and Methods of the Pablo SDDF Interface Library

The Picasso Research Group
University of Illinois at Urbana-Champaign
Department of Computer Science
1304 West Springfield Avenue
Urbana, Illinois 61801

April 11, 1994

Copyright © 1992-1994 The University of Illinois Board of Trustees. All Rights Reserved.
Pablo is a registered trademark of the Board of Trustees of the University of Illinois.

CONFIDENTIAL INFORMATION.
Distribution restricted under license agreement.

Supported in part by the Advanced Research Projects Agency under ARPA Contract Number DABT63-91-K-0004, by the National Science Foundation under grants NSF CCR87-06653 and NSF CDA87-22836, by the National Aeronautics and Space Administration under NASA Contract Number NAG-1-613, by a grant from the Digital Equipment Corporation External Research Program, and by a collaborative research agreement with the Intel Supercomputer Systems Division.

Name

Array – Class used to represent and manipulate the primitive type for variable-length, multidimensional, single-typed arrays.

Authors

Ruth A. Aydt (aydt@cs.uiuc.edu)
Contributing Author: Daniel A. Reed (reed@cs.uiuc.edu)
Contributing Author: Brian K. Totty (totty@cs.uiuc.edu)

Description

The Array Class is used to represent and manipulate data values in variable-length, multidimensional, single-typed arrays. Once an Array instance is created the only way it's type or it's dimension can be changed is with the = operator.

Available Methods

Public

- atomSize – Return the sizeof() for an atom of the specified type.
- bitsToObj – Use information in the binary bitstream to fill in number of cells in each dimension and cell values. Return the number of bytes read or -1 if an error occured.
- bytesNeeded – Return number of bytes needed to represent Array as a bitstream
- cbitsToObj – Use information in the non-native binary bitstream to fill in number of cells in each dimension and cell values. Return the number of bytes read or -1 if an error occured.
- cbytesNeeded – Return number of bytes needed to represent Array as a non-native binary bitstream
- clearDelta – Sets the number of delta array elements to zero
- getCellCount – Return total number of cells in array.
- getCellValue – Return value of specified cell. General method for arrays with >2 dimensions.
- getCellValue – Return value of specified cell. Specialized method for 2-d arrays to enhance performance.
- getCellValue – Return value of specified cell. Specialized method for vectors to enhance performance.
- getCellValue – Update value parameter with contents of of specified cell. Special case for 2-d arrays. Returns FAILURE_ if error such as index out-of-bounds.
- getCellValue – Update value parameter with contents of specified cell. General case for >2-d arrays. Returns FAILURE_ if error such as index out-of-bounds.
- getCellValue – Update value parameter with contents of specified cell. Special case for vectors. Returns FAILURE_ if error such as index out- of-bounds.
- getDeltaCount – Return the number of changed cells in the delta array
- getDeltaValue – Returns both the next delta value and its array indices

- getDimSizes – Return pointer to integer array specifing the number of cells in each dimension
- getDimension – Return the dimension of the array
- getTheCellValue – Return value of the cell, where index is the absolute index into the array rather than an index into each dimension.
- getTheCellValue – Update value parameter with the contents of the cell, where index is the absolute index into the array rather than an index into each dimension. Return FAILURE_ if error such as index out-of-bounds.
- getType – Return type of array values
- isDeltaArray – Returns TRUE_ if the array was constructed to use deltas (i.e., shows changes)
- objToBits – Represent the dimension sizes and data as bits in the buffer pointed to by bufPtr. Return the number of bytes used or -1 if the buffer was too small.
- objToCbits – Represent the dimension sizes and data as bits as a non-native binary bitstream in the buffer pointed to by bufPtr. Return the number of bytes used or -1 if the buffer was too small.
- operator= – Array to Array assignment. Upon completion, both Array instances have copies of the data. If this Array has different type, dimension, or dimension sizes than the new, we release the old space and alloc the new. Else, we reuse the space.
- printOn – Helper function for Array output
- resetDelta – Resets the number of changed delta elements to the specified value
- resetIterateCount – Resets the number of iterates fetched
- setCellString – Set the cells beginning at the specified offset to the characters in the string argument. If the Array had dimension N, then there should be N-1 index arguments in the call. FAILURE_ is returned if this is not a CHARACTER array, if the index values are bogus, or if dimSizes have not yet been set. If the string is longer than those supported by the array - whatever dimSizes[N-1] equals - then the string is truncated.
- setCellValue – Set specified cell to given value. The type of the array will NOT change - the value will be recast as needed. Returns FAILURE_ if index out of bounds or if dimension sizes of array not yet set.
- setCellValues – Set all cells without changing the current array type. Data are recast as needed. FAILURE_ is returned if the array has not yet been sized, or if the new array had different dimension or dimension sizes than this array.
- setDeltaIndex – Updates the list of deltas to indicate that the value at the specified index has been changed
- setDimSizes – Set number of cells in each dimension from the array of integers pointed to by intArray. The data area is resized and initialized to 0.
- setTheCellValue – Set cell to given value, where the absolute index into the array is given rather than an index for each dimension. The type of the array will NOT change - the value will be recast if needed. Returns FAILURE_ if index out of bounds or if dimension sizes of array not yet set.

Included Files

"DataTraits.h"

"Obj.h"

Public Methods

Array(`MachineDataType _type, int _dimension, Boolean_ _useDelta = FALSE_ );`

Constructor with type, dimension, and the potential use of array deltas specified. No space is allocated for the array cells

Array(`const Array& array );`

Constructor with copy initialization

Array(`const Array *arrayP );`

Constructor with pointer to existing Array instance as an argument.

~Array(`);`

The Destructor.

`int` **atomSize(** `MachineDataType atomType ) const;`

Return the sizeof() for an atom of the specified type.

`int` **bitsToObj(** `const char *bufPtr );`

Use information in the binary bitstream to fill in number of cells in each dimension and cell values. Return the number of bytes read or -1 if an error occured.

`int` **bytesNeeded(** `) const;`

Return number of bytes needed to represent Array as a bitstream

`int` **cbitsToObj(** `const char *bufPtr, const DataCharacteristics *cnvInfo );`

Use information in the non-native binary bitstream to fill in number of cells in each dimension and cell values. Return the number of bytes read or -1 if an error occured.

`int` **cbytesNeeded(** `const DataCharacteristics *cnvInfo ) const;`

Return number of bytes needed to represent Array as a non-native binary bitstream

`void` **clearDelta(** `);`

Sets the number of delta array elements to zero

`int` **getCellCount(** `) const;`

Return total number of cells in array.

`Value` **getCellValue**`( int dim0Index ) const;`

Return value of specified cell. Specialized method for vectors to enhance performance.

`Value` **getCellValue**`( int dim0Index, int dim1Index ) const;`

Return value of specified cell. Specialized method for 2-d arrays to enhance performance.

`Value` **getCellValue**`( int dim0Index, int dim1Index, int dim2Index ... ) const;`

Return value of specified cell. General method for arrays with >2 dimensions.

`Boolean_` **getCellValue**`( Value& value, int dim0Index ) const;`

Update value parameter with contents of specified cell. Special case for vectors. Returns FAILURE_ if error such as index out- of-bounds.

`Boolean_` **getCellValue**`( Value& value, int dim0Index, int dim1Index ) const;`

Update value parameter with contents of of specified cell. Special case for 2-d arrays. Returns FAILURE_ if error such as index out-of-bounds.

`Boolean_` **getCellValue**`( Value& value, int dim0Index, int dim1Index,`
`int dim2Index ... ) const;`

Update value parameter with contents of specified cell. General case for >2-d arrays. Returns FAILURE_ if error such as index out-of-bounds.

`int` **getDeltaCount**`( ) const;`

Return the number of changed cells in the delta array

`void` **getDeltaValue**`( Value& value, int* dimVector );`

Returns both the next delta value and its array indices

`const int *` **getDimSizes**`( ) const;`

Return pointer to integer array specifing the number of cells in each dimension

`int` **getDimension**`( ) const;`

Return the dimension of the array

`MachineDataType` **getType**`( ) const;`

Return type of array values

`Value` **getTheCellValue**`( int index ) const;`

Return value of the cell, where index is the absolute index into the array rather than an index into each dimension.

`Boolean_` **getTheCellValue**`( Value& value, int index ) const;`

Update value parameter with the contents of the cell, where index is the absolute index into the array rather than an index into each dimension. Return FAILURE_ if error such as index out-of-bounds.

`Boolean_` **isDeltaArray**`( ) const;`

Returns TRUE_ if the array was constructed to use deltas (i.e., shows changes)

`int` **objToBits**`( char *const bufPtr, int bufLen ) const;`

Represent the dimension sizes and data as bits in the buffer pointed to by bufPtr. Return the number of bytes used or -1 if the buffer was too small.

`int` **objToCbits**`( char *const bufPtr, int bufLen,`
`const DataCharacteristics *cnvInfo ) const;`

Represent the dimension sizes and data as bits as a non-native binary bitstream in the buffer pointed to by bufPtr. Return the number of bytes used or -1 if the buffer was too small.

`void` **resetIterateCount**`( );`

Resets the number of iterates fetched

`void` **resetDelta**`( int countDelta );`

Resets the number of changed delta elements to the specified value

`Boolean_` **setCellString**`( const CString& cstring, int dim0Index = 0 ... );`

Set the cells beginning at the specified offset to the characters in the string argument. If the Array had dimension N, then there should be N-1 index arguments in the call. FAILURE_ is returned if this is not a CHARACTER array, if the index values are bogus, or if dimSizes have not yet been set. If the string is longer than those supported by the array - whatever dimSizes[N-1] equals - then the string is truncated.

`Boolean_` **setCellValue**`( const Value& value, int dim0Index ... );`

Set specified cell to given value. The type of the array will NOT change - the value will be recast as needed. Returns FAILURE_ if index out of bounds or if dimension sizes of array not yet set.

`Boolean_` **setCellValues**`( const Array * array );`

Set all cells without changing the current array type. Data are recast as needed. FAILURE_ is returned if the array has not yet been sized, or if the new array had different dimension or dimension sizes than this array.

`void` **setDimSizes**`( const int *intArray );`

Set number of cells in each dimension from the array of integers pointed to by intArray. The data area is resized and initialized to 0.

`Boolean_` **setTheCellValue**`( int index, const Value& value );`

Set cell to given value, where the absolute index into the array is given rather than an index for each dimension. The type of the array will NOT change - the value will be recast if needed. Returns FAILURE_ if index out of bounds or if dimension sizes of array not yet set.

`void` **setDeltaIndex**`( int deltaIndex );`

Updates the list of deltas to indicate that the value at the specified index has been changed

`Array&` **operator**`=( const Array& array );`

Array to Array assignment. Upon completion, both Array instances have copies of the data. If this Array has different type, dimension, or dimension sizes than the new, we release the old space and alloc the new. Else, we reuse the space.

`void` **printOn**`( ostream& strm = cout ) const;`

Helper function for Array output

Name

AsciiPipeReader – class dealing with input from an ascii pipe

Authors

Ruth A. Aydt (aydt@cs.uiuc.edu)
Contributing Author: Tom Birkett (birkett@cs.uiuc.edu)
Contributing Author: Bobby A.A. Nazief (nazief@cs.uiuc.edu)

Heritage

Inherits from – PipeReader, Obj.

Description

The AsciiPipeReader class contains methods designed to accept input from an ascii pipe.

Available Methods

Private

- _badInput – Call abort() with a message indicating what was expected and where the parsing failed. You never return from this method.
- _getArray – Parse an Array data record picking out dimension and cell values.
- _getAttributes – Parse data and fill in attributes with key/value pairs found.
- _getCommandTag – Parse data and return integer tag for the command packet.
- _getDataTag – Parse data for the name and look up tag in the nameToTag table. Tag is returned.
- _getDescriptorTag – Parse data and return the integer tag for the descriptor. Also insert the name/tag in the nameToTag table.
- _getField – Parse data for the field information and set fieldDescr values.
- _getPipeAttributes – Parse data for the attributes and insert key/value pairs in pipeAttributes.
- _getString – Parse data and string.
- _getType – Parse data and set type.
- _getValue – Parse data for a value.
- _readPacket – Read the next packet from the pipe. Update currentHeader & pipeAttributes
- _skipWhiteSpace – Skip over white space in data.

Protected

- _setClassName (inherited) – Set _className

Public

- abort (inherited) – Print message and abort system.
- error (inherited) – This prints an error message.
- gentleWarning (inherited) – A warning message to stderr

- getAttributes (inherited) – Copy pipeAttributes to attributes
- getClassName (inherited) – Return pointer to class name
- getData – Parse the record data in internal buffer into recordDossier.
- getDescriptor – Parse the structure descriptor in internal buffer into structDescr.
- getFileSize (inherited) – Returns the size of the file the pipe is connected to. Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- getObjectName (inherited) – This returns a name for the object. As defined in Obj, it returns the same string as the is returned by getClassName. However, derived classes may redefine it to include more detailed information.
- getOffset (inherited) – Returns the current offset into the pipe being read from. Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- getPacketDataPtr – Virtual method that does not make sense for this class & is not implemented.
- getPacketHeader – Get header of packet next in pipe. If packet is there it is actually read into packetBuffer.
- hash (inherited) – Return a hash value for this object
- info (inherited) – Prints informational message.
- inspect (inherited) – Inspect the object.
- notImplemented (inherited) – Print an error message that the argument list is not implemented
- printOn (inherited) – Helper function for output
- rewind (inherited) – Rewind the offset to what it was when the PipeReader was created. Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- setOffset (inherited) – Sets the current offset into the pipe being read from. Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- shouldNotImplement (inherited) – This method prints an error message that says it should not be implemented.
- subclassResponsibility (inherited) – Method prints an error message that says it should be implemented by a subclass.
- successfulConnection (inherited) – Returns TRUE_ if the connection was successful in the constructor.
- warning (inherited) – This prints a warning message.

Included Files

"DataTraits.h"
"PipeReader.h"

Private Methods

`void` **_badInput(** `const char *message, const char *parseString ) const;`

Call abort() with a message indicating what was expected and where the parsing failed. You never return from this method.

`void` **_getArray**`( Array *array, MachineDataType type, int dimension );`

Parse an Array data record picking out dimension and cell values.

`void` **_getAttributes**`( BaseDescriptor & descriptor );`

Parse data and fill in attributes with key/value pairs found.

`int` **_getCommandTag**`( );`

Parse data and return integer tag for the command packet.

`int` **_getDataTag**`( );`

Parse data for the name and look up tag in the nameToTag table. Tag is returned.

`int` **_getDescriptorTag**`( );`

Parse data and return the integer tag for the descriptor. Also insert the name/tag in the nameToTag table.

`void` **_getField**`( FieldDescriptor& fieldDescr );`

Parse data for the field information and set fieldDescr values.

`void` **_getPipeAttributes**`( );`

Parse data for the attributes and insert key/value pairs in pipeAttributes.

`void` **_getString**`( char **string );`

Parse data and string.

`void` **_getType**`( MachineDataType& type );`

Parse data and set type.

`void` **_getValue**`( Value& val, MachineDataType type );`

Parse data for a value.

`Boolean_` **_readPacket**`( );`

Read the next packet from the pipe. Update currentHeader & pipeAttributes

`void` **_skipWhiteSpace**`( );`

Skip over white space in data.

Public Methods

AsciiPipeReader(`StreamPipe *initPipe` **);**

The constructor.

~AsciiPipeReader(**);**

The destructor.

`virtual Boolean_` **getData(** `RecordDossier& recordDossier` **);**

Parse the record data in internal buffer into recordDossier.

`virtual Boolean_` **getDescriptor(** `StructureDescriptor& structDescr` **);**

Parse the structure descriptor in internal buffer into structDescr.

`virtual const char *` **getPacketDataPtr(** **)** `const ;`

Virtual method that does not make sense for this class & is not implemented.

`virtual PacketHeader` **getPacketHeader(** **);**

Get header of packet next in pipe. If packet is there it is actually read into packetBuffer.

Name

AsciiPipeWriter – Outputs a Ascii data file given this file's internal representation.

Authors

Ruth A. Aydt (aydt@cs.uiuc.edu)
Contributing Author: Tom Birkett (birkett@cs.uiuc.edu)
Contributing Author: Bobby A.A. Nazief (nazief@cs.uiuc.edu)

Heritage

Inherits from – PipeWriter, Obj.

Description

An AsciiPipeWriter is the class which produces the Ascii file representation from the internal representation of the data and the data's description.

Available Methods

Private
- _checkBufferSize – Make sure there is space for cnt characters in packetBuffer and if not call _increaseBufferSize()
- _increaseBufferSize – If a newSize is nonzero, increase the size of packetBuffer to that. Else, increase it by some fixed value
- _putArray – Format the array values and write the to packetBuffer.
- _putC – Write the character to packetBuffer and increment bufp
- _putValue – Format the value and write it to packetBuffer.
- _writeCString – Write the CString to the buffer and enclose it in ""'s.
- _writePacket – Write the packet constructed in packetBuffer to the pipe.

Protected
- _setClassName (inherited) – Set _className
- _writeSddfHeader – Write the SDDF header information to the current Stream Pipe.

Public
- abort (inherited) – Print message and abort system.
- addPipe – Add another pipe to the list we write to.
- deletePipe (inherited) – Delete the pipe from the list we write to.
- error (inherited) – This prints an error message.
- gentleWarning (inherited) – A warning message to stderr
- getClassName (inherited) – Return pointer to class name
- getFileSize (inherited) – Returns the size of the file the pipe is connected to. Not valid for all types of pipes - if call made and invalid, program aborts with error message.

- getObjectName (inherited) – This returns a name for the object. As defined in Obj, it returns the same string as the is returned by getClassName. However, derived classes may redefine it to include more detailed information.
- getOffset (inherited) – Returns the current offset into the pipe being written to. Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- hash (inherited) – Return a hash value for this object
- info (inherited) – Prints informational message.
- inspect (inherited) – Inspect the object.
- notImplemented (inherited) – Print an error message that the argument list is not implemented
- printOn (inherited) – Stream output function
- putAttributes – Update pipeAttributes and write them to the pipe.
- putCommand – Write a Command packet with the given tag to the pipe
- putData – Write the data associated with a record to the pipe.
- putDescriptor – Write the structure Descriptor to the pipe.
- putPacket – Virtual method that does not make sense for this class and is not implemented.
- rewind (inherited) – Rewind the offset to what it was when the PipeWriter was created and the header written (file is truncated to the shorter length). Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- setOffset (inherited) – Sets the current offset into the pipe being read from (no change in data takes place). Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- shouldNotImplement (inherited) – This method prints an error message that says it should not be implemented.
- subclassResponsibility (inherited) – Method prints an error message that says it should be implemented by a subclass.
- successfulConnection (inherited) – Returns TRUE_ if the connection was successful in the constructor.
- warning (inherited) – This prints a warning message.

Included Files

"PipeWriter.h"

Private Methods

`void` **_checkBufferSize**`(  int cnt );`

Make sure there is space for cnt characters in packetBuffer and if not call _increaseBufferSize()

`void` **_increaseBufferSize**`(  int newSize = 0 );`

If a newSize is nonzero, increase the size of packetBuffer to that. Else, increase it by some fixed value

`void` **_putArray**`(  Array *arrayPtr );`

Format the array values and write the to packetBuffer.

`void` **_putC**`( char c );`

Write the character to packetBuffer and increment bufp

`void` **_putValue**`( Value *valuePtr );`

Format the value and write it to packetBuffer.

`void` **_writeCString**`( const CString& cstr );`

Write the CString to the buffer and enclose it in ""'s.

`void` **_writePacket**`( );`

Write the packet constructed in packetBuffer to the pipe.

Protected Methods

`virtual void` **_writeSddfHeader**`( ) const;`

Write the SDDF header information to the current Stream Pipe.

Public Methods

AsciiPipeWriter`( StreamPipe *thePipe );`

The constructor

~**AsciiPipeWriter**`( );`

The destructor

`virtual Boolean_` **addPipe**`( StreamPipe *thePipe );`

Add another pipe to the list we write to.

`virtual void` **putAttributes**`( );`

Write pipeAttributes to the pipe

`virtual void` **putAttributes**`( const Attributes& attributes );`

Update pipeAttributes and write them to the pipe.

`virtual void` **putCommand**`( int pktTag );`

Write a Command packet with the given tag to the pipe

`virtual void` **putData(** `const RecordDossier& recordDossier );`

Write the data associated with a record to the pipe.

`virtual void` **putDescriptor(** `const StructureDescriptor& structDescr, int pktTag );`

Write the structure Descriptor to the pipe.

`virtual void` **putPacket(** `const PacketHeader&, const char * ) ;`

Virtual method that does not make sense for this class and is not implemented.

Name

Attributes – Linked list implementation class for key/value pairs.

Authors

Daniel A. Reed (reed@cs.uiuc.edu)
Contributing Author: Ruth A. Aydt (aydt@cs.uiuc.edu)

Heritage

Inherits from – Obj.

Description

Attributes is the implementation class for associative data structures (i.e., item retrieval by key rather than index).

Friends

Class AttributesIterator.

Available Methods

Protected

- _setClassName (inherited) – Set _className

Public

- abort (inherited) – Print message and abort system.
- bitsToObj – Method used to translate from binary bitstream representation of data to Attributes instance. Any key/value pairs existing in the Attributes list when the call is made are removed. Returns number of bytes read or -1 if some error occured.
- bytesNeeded – Method which returns the number of bytes needed to represent the Attributes Object instance as a binary bitstream.
- cbitsToObj – Method used to translate from non-native binary bitstream representation of data to Attributes instance. Any key/value pairs existing in the Attributes list when the call is made are removed. Returns number of bytes read or -1 if some error occured.
- cbytesNeeded – Method which returns the number of bytes needed to represent the Attributes Object instance as a non-native binary bitstream.
- clearEntries – Method to remove all key/value entries from the Attributes list
- contains – Attributes membership test
- entryCount – Number of key/value entries
- error (inherited) – This prints an error message.

- fetch – Value extraction based on key. Returns value or NOMATCH if no match.
- gentleWarning (inherited) – A warning message to stderr
- getClassName (inherited) – Return pointer to class name
- getObjectName (inherited) – This returns a name for the object. As defined in Obj, it returns the same string as the is returned by getClassName. However, derived classes may redefine it to include more detailed information.
- hash (inherited) – Return a hash value for this object
- info (inherited) – Prints informational message.
- insert – Add key/value pair to list. Returns SUCCESS_ or FAILURE_ (if duplicate key).
- inspect (inherited) – Inspect the object.
- notImplemented (inherited) – Print an error message that the argument list is not implemented
- objToBits – Method used to translate internal Attributes instance to bitstream representation of data. Returns number of bytes used to write bitstream data, or -1 if bufLen was less than the number of bytes needed to represent data in binary bitstream format.
- objToCbits – Method used to translate internal Attributes instance to non-native bitstream representation of data. Returns number of bytes used to write bitstream data, or -1 if bufLen was less than the number of bytes needed to represent data in bitstream format.
- operator!= – Attributes non-equivalence test
- operator= – Attributes assignment (with copies)
- operator== – Attributes equivalence test. Attributes are considered equivalent if they have identical key/value pairs appearing in the Attributes list IN THE SAME ORDER
- operator[] – Attributes extraction based on key
- printOn – helper function for Attributes output
- remove – Remove key/value pair from list
- shouldNotImplement (inherited) – This method prints an error message that says it should not be implemented.
- subclassResponsibility (inherited) – Method prints an error message that says it should be implemented by a subclass.
- warning (inherited) – This prints a warning message.

Included Files

"CString.h"
"Obj.h"

Public Methods

Attributes( );

The simplest Attributes constructor

Attributes(const Attributes& attributes);

Attributes constructor with copy initialization

~Attributes(**);**

The Attributes destructor

`int` **bitsToObj(** `const char *bufPtr );`

Method used to translate from binary bitstream representation of data to Attributes instance. Any key/value pairs existing in the Attributes list when the call is made are removed. Returns number of bytes read or -1 if some error occured.

`int` **bytesNeeded(** `) const;`

Method which returns the number of bytes needed to represent the Attributes Object instance as a binary bitstream.

`int` **cbitsToObj(** `const char *bufPtr, const DataCharacteristics *cnvInfo );`

Method used to translate from non-native binary bitstream representation of data to Attributes instance. Any key/value pairs existing in the Attributes list when the call is made are removed. Returns number of bytes read or -1 if some error occured.

`int` **cbytesNeeded(** `const DataCharacteristics *cnvInfo ) const;`

Method which returns the number of bytes needed to represent the Attributes Object instance as a non-native binary bitstream.

`void` **clearEntries(** `);`

Method to remove all key/value entries from the Attributes list

`Boolean_` **contains(** `const CString& key ) const;`

Attributes membership test

`int` **entryCount(** `) const;`

Number of key/value entries

`const CString&` **fetch(** `const CString& key ) const;`

Value extraction based on key. Returns value or NOMATCH if no match.

`Boolean_` **insert(** `const CString& key, const CString& value );`

Add key/value pair to list. Returns SUCCESS_ or FAILURE_ (if duplicate key).

`int` **objToBits(** `char *const bufPtr, int bufLen ) const;`

Method used to translate internal Attributes instance to bitstream representation of data. Returns number of bytes used to write bitstream data, or -1 if bufLen was less than the number of bytes needed to represent data in binary bitstream format.

```
int objToCbits(   char *const bufPtr,  int bufLen,
                  const DataCharacteristics *cnvInfo ) const;
```

Method used to translate internal Attributes instance to non-native bitstream representation of data. Returns number of bytes used to write bitstream data, or -1 if bufLen was less than the number of bytes needed to represent data in bitstream format.

```
void remove(   const CString& key );
```

Remove key/value pair from list

```
const CString& operator[](   const CString& key ) const;
```

Attributes extraction based on key

```
Attributes& operator=(   const Attributes& attributes );
```

Attributes assignment (with copies)

```
Boolean_ operator==(   const Attributes& attributes ) const;
```

Attributes equivalence test. Attributes are considered equivalent if they have identical key/value pairs appearing in the Attributes list IN THE SAME ORDER

```
Boolean_ operator!=(   const Attributes& attributes ) const;
```

Attributes non-equivalence test

```
virtual void printOn(   ostream& strm = cout ) const;
```

helper function for Attributes output

Name

AttributesIterator – Class used to step through keys in an Attributes instance.

Authors

Ruth A. Aydt (aydt.cs.uiuc.edu)

Heritage

Inherits from – Obj.

Description

An AttributesIterator is used to step through the keys of entries in an Attributes instance. It's member variables maintain a sense of the "current" Entry in the Attributes instance.

Available Methods

Protected

- _setClassName (inherited) – Set _className

Public

- abort (inherited) – Print message and abort system.
- error (inherited) – This prints an error message.
- first – Returns reference to key of first Entry in the Attributes list or to NOMATCH if there isn't one.
- gentleWarning (inherited) – A warning message to stderr
- getClassName (inherited) – Return pointer to class name
- getObjectName (inherited) – This returns a name for the object. As defined in Obj, it returns the same string as the is returned by getClassName. However, derived classes may redefine it to include more detailed information.
- hash (inherited) – Return a hash value for this object
- info (inherited) – Prints informational message.
- inspect (inherited) – Inspect the object.
- next – Returns reference to key of the next entry in the Attributes list or to NOMATCH if there isn't one. If called before first() for a particular instance of the iterator it behaves the same as first().
- notImplemented (inherited) – Print an error message that the argument list is not implemented
- printOn (inherited) – Stream output function
- shouldNotImplement (inherited) – This method prints an error message that says it should not be implemented.
- subclassResponsibility (inherited) – Method prints an error message that says it should be implemented by a subclass.

- theValue – Returns reference to value of the "current" entry in the Attributes list or to NOMATCH if there in no valid current entry
- warning (inherited) – This prints a warning message.

Included Files

"Obj.h"
"Attributes.h"

Public Methods

AttributesIterator(`const Attributes& attributes` **);**

The Iterator constructor with reference to Attributes.

~AttributesIterator(**);**

The Iterator destructor

`const CString&` **first(** **);**

Returns reference to key of first Entry in the Attributes list or to NOMATCH if there isn't one.

`const CString&` **next(** **);**

Returns reference to key of the next entry in the Attributes list or to NOMATCH if there isn't one. If called before first() for a particular instance of the iterator it behaves the same as first().

`const CString&` **theValue(** **);**

Returns reference to value of the "current" entry in the Attributes list or to NOMATCH if there in no valid current entry

Name

BaseDescriptor – Base class for self-documenting data format component. Contains name and attribute information and methods.

Authors

Ruth A. Aydt (aydt@cs.uiuc.edu)

Heritage

Inherits from – Obj.

Description

A BaseDescriptor defines the base class for the component descriptors in the self documenting data format. You will never have an instance of this class. It contains the component name, attributes, and defines the associated methods.

Available Methods

Protected

- _setClassName (inherited) – Set _className

Public

- abort (inherited) – Print message and abort system.
- error (inherited) – This prints an error message.
- gentleWarning (inherited) – A warning message to stderr
- getAttributes – Returns the attribute dictionary for this component
- getClassName (inherited) – Return pointer to class name
- getName – Returns the name of the component
- getObjectName – Overloads Obj::getObjectName() to return the full name including class and component names.
- hash (inherited) – Return a hash value for this object
- info (inherited) – Prints informational message.
- insertAttributes – Insert a new attribute
- inspect (inherited) – Inspect the object.
- notImplemented (inherited) – Print an error message that the argument list is not implemented
- notNull – Returns FALSE_ if name is NOMATCH, otherwise returns TRUE_.
- printOn – Helper function for Descriptor output. Must be redefined by subclasses.
- setAttributes – Set the attribute dictionary for this component
- setName – Set the name of the component. Also update objectName;
- shouldNotImplement (inherited) – This method prints an error message that says it should not be implemented.

- subclassResponsibility (inherited) – Method prints an error message that says it should be implemented by a subclass.
- warning (inherited) – This prints a warning message.

Included Files

"Attributes.h"
"CString.h"
"Obj.h"

Public Methods

BaseDescriptor( );
The constructor.

~BaseDescriptor( );
The destructor.

`virtual const char * getObjectName( ) const;`
Overloads Obj::getObjectName() to return the full name including class and component names.

`const Attributes& getAttributes( ) const;`
Returns the attribute dictionary for this component

`const CString& getName( ) const;`
Returns the name of the component

`Boolean_ insertAttributes( const CString& key, const CString& value );`
Insert a new attribute

`Boolean_ notNull( ) const;`
Returns FALSE_ if name is NOMATCH, otherwise returns TRUE_.

`void setAttributes( const Attributes& componentAttributes );`
Set the attribute dictionary for this component

`void setName( const CString& componentName );`
Set the name of the component. Also update objectName;

`virtual void printOn( ostream& )const ;`
Helper function for Descriptor output. Must be redefined by subclasses.

Name

BasePipe – Base class for pipes

Authors

Robert Olson (olson@cs.uiuc.edu)

Heritage

Inherits from – Obj.

Description

BasePipe is the class from which all Pipes are derived. It has support for connections to wrappers. The methods for reading and writing from Pipes are defined in the two major subclasses, StreamPipe and BlockPipe. The object name is set to be the name in the constructor

Available Methods

Protected
- _setClassName (inherited) – Set _className
- _setObjectName – Give detailed name to Pipe

Public
- abort (inherited) – Print message and abort system.
- error (inherited) – This prints an error message.
- gentleWarning (inherited) – A warning message to stderr
- getClassName (inherited) – Return pointer to class name
- getInputWrapper – Get the input wrapper
- getObjectName – Overloads Obj::getObjectName() to return _objectName
- getOutputWrapper – Get the output wrapper
- hash (inherited) – Return a hash value for this object
- info (inherited) – Prints informational message.
- inspect (inherited) – Inspect the object.
- isFilePipe – Returns TRUE_ if this is a file pipe
- isInputPipe – Returns TRUE_ if this is an input pipe
- isOutputPipe – Returns TRUE_ if this is an output pipe
- notImplemented (inherited) – Print an error message that the argument list is not implemented
- printOn – Stream output function
- setInputWrapper – Set the input wrapper
- setOutputWrapper – Set the output wrapper
- shouldNotImplement (inherited) – This method prints an error message that says it should not be implemented.

- subclassResponsibility (inherited) – Method prints an error message that says it should be implemented by a subclass.
- warning (inherited) – This prints a warning message.

Included Files

"Obj.h"

Protected Methods

`void` **_setObjectName**`( const char *detailName );`

Give detailed name to Pipe

Public Methods

BasePipe`( );`

The BasePipe constructor

~**BasePipe**`( );`

The BasePipe destructor

`virtual const char *` **getObjectName**`( ) const;`

Overloads Obj::getObjectName() to return _objectName

`virtual Boolean_` **isFilePipe**`( ) const ;`

Returns TRUE_ if this is a file pipe

`virtual Boolean_` **isInputPipe**`( ) const ;`

Returns TRUE_ if this is an input pipe

`virtual Boolean_` **isOutputPipe**`( ) const ;`

Returns TRUE_ if this is an output pipe

`Wrapper *` **getInputWrapper**`( ) const;`

Get the input wrapper

`Wrapper *` **getOutputWrapper**`( ) const;`

Get the output wrapper

`void` **setInputWrapper(** `Wrapper *iw );`

Set the input wrapper

`void` **setOutputWrapper(** `Wrapper *ow );`

Set the output wrapper

`virtual void` **printOn(** `ostream& os) const;`

Stream output function

Name

BinaryPipeReader – A class designed to handle an input from a binary pipe.

Authors

Ruth A. Aydt (aydt@cs.uiuc.edu)
Contributing Author: Tom Birkett (birkett@cs.uiuc.edu)
Contributing Author: Taed Nelson (nelson@cs.uiuc.edu)

Heritage

Inherits from – PipeReader, Obj.

Description

The BinaryPipeReader class implements methods which handle reading input from a binary pipe. This is accomplished primarily by calling the appropriate parser methods in the classes corresponding to the sort of data we are reading (Descriptors, Attributes, ...).

Available Methods

Private
- _readPacket – Grabs a packet from the pipe, updating currentHeader and pipeAttributes as appropriate

Protected
- _setClassName (inherited) – Set _className

Public
- abort (inherited) – Print message and abort system.
- error (inherited) – This prints an error message.
- gentleWarning (inherited) – A warning message to stderr
- getAttributes (inherited) – Copy pipeAttributes to attributes
- getClassName (inherited) – Return pointer to class name
- getData – Parse the record data in interal buffer into recordDossier.
- getDescriptor – Parse the structure descriptor in internal buffer into structDescr
- getFileSize (inherited) – Returns the size of the file the pipe is connected to. Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- getObjectName (inherited) – This returns a name for the object. As defined in Obj, it returns the same string as the is returned by getClassName. However, derived classes may redefine it to include more detailed information.
- getOffset (inherited) – Returns the current offset into the pipe being read from. Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- getPacketDataPtr – Return pointer to data section of packetBuffer

- getPacketHeader – Get header of packet next in pipe. If packet is there, it is actually read into packetBuffer
- hash (inherited) – Return a hash value for this object
- info (inherited) – Prints informational message.
- inspect (inherited) – Inspect the object.
- notImplemented (inherited) – Print an error message that the argument list is not implemented
- printOn (inherited) – Helper function for output
- rewind (inherited) – Rewind the offset to what it was when the PipeReader was created. Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- setOffset (inherited) – Sets the current offset into the pipe being read from. Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- shouldNotImplement (inherited) – This method prints an error message that says it should not be implemented.
- subclassResponsibility (inherited) – Method prints an error message that says it should be implemented by a subclass.
- successfulConnection (inherited) – Returns TRUE_ if the connection was successful in the constructor.
- warning (inherited) – This prints a warning message.

Included Files

"Defines.h"
"PipeReader.h"

Private Methods

`Boolean_ _readPacket( );`

Grabs a packet from the pipe, updating currentHeader and pipeAttributes as appropriate

Public Methods

BinaryPipeReader(`StreamPipe *initPipe );`

The constructor

~BinaryPipeReader(`);`

The destructor

`virtual Boolean_` **getData(** `RecordDossier& recordDossier );`

Parse the record data in interal buffer into recordDossier.

`virtual Boolean_` **getDescriptor(** `StructureDescriptor& structDescr );`

Parse the structure descriptor in internal buffer into structDescr

`virtual const char * getPacketDataPtr( ) const;`

Return pointer to data section of packetBuffer

`PacketHeader getPacketHeader( );`

Get header of packet next in pipe. If packet is there, it is actually read into packetBuffer

Name

BinaryPipeWriter – Outputs binary data

Authors

Ruth A. Aydt (aydt@cs.uiuc.edu)
Contributing Author: Bobby A.A. Nazief (nazief@cs.uiuc.edu)
Contributing Author: Taed Nelson (nelson@cs.uiuc.edu)

Heritage

Inherits from – PipeWriter, Obj.

Description

BinaryPipeWriter is the class which writes data to a binary pipe.

Available Methods

Private

- _writePacket – Write the packet constructed in packetBuffer to the pipe.

Protected

- _setClassName (inherited) – Set _className
- _writeSddfHeader – Write the SDDF header information to the current Stream Pipe.

Public

- abort (inherited) – Print message and abort system.
- addPipe (inherited) – Add another pipe to the list we write to.
- deletePipe (inherited) – Delete the pipe from the list we write to.
- error (inherited) – This prints an error message.
- gentleWarning (inherited) – A warning message to stderr
- getClassName (inherited) – Return pointer to class name
- getFileSize (inherited) – Returns the size of the file the pipe is connected to. Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- getObjectName (inherited) – This returns a name for the object. As defined in Obj, it returns the same string as the is returned by getClassName. However, derived classes may redefine it to include more detailed information.
- getOffset (inherited) – Returns the current offset into the pipe being written to. Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- hash (inherited) – Return a hash value for this object
- info (inherited) – Prints informational message.
- inspect (inherited) – Inspect the object.
- notImplemented (inherited) – Print an error message that the argument list is not implemented

- printOn (inherited) – Stream output function
- putAttributes – Update pipeAttributes and write them to the pipe.
- putCommand – Write the command to the pipe
- putData – Write the data associated with a record to the pipe.
- putDescriptor – Write the structure Descriptor to the pipe.
- putPacket – Write the packet header and data (if it exists) to the pipe
- rewind (inherited) – Rewind the offset to what it was when the PipeWriter was created and the header written (file is truncated to the shorter length). Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- setOffset (inherited) – Sets the current offset into the pipe being read from (no change in data takes place). Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- shouldNotImplement (inherited) – This method prints an error message that says it should not be implemented.
- subclassResponsibility (inherited) – Method prints an error message that says it should be implemented by a subclass.
- successfulConnection (inherited) – Returns TRUE_ if the connection was successful in the constructor.
- warning (inherited) – This prints a warning message.

Included Files

"PipeWriter.h"

Private Methods

`void` **_writePacket**`( );`

Write the packet constructed in packetBuffer to the pipe.

Protected Methods

`virtual void` **_writeSddfHeader**`( ) const;`

Write the SDDF header information to the current Stream Pipe.

Public Methods

BinaryPipeWriter`( StreamPipe *thePipe );`

The constructor

~BinaryPipeWriter`( );`

The destructor

`virtual void` **putAttributes(** `);`

Write pipeAttributes to the pipe

`virtual void` **putAttributes(** `const Attributes& attributes );`

Update pipeAttributes and write them to the pipe.

`virtual void` **putCommand(** `int pktTag = 0 );`

Write the command to the pipe

`virtual void` **putData(** `const RecordDossier& recordDossier );`

Write the data associated with a record to the pipe.

`virtual void` **putDescriptor(** `const StructureDescriptor& structDescr, int pktTag );`

Write the structure Descriptor to the pipe.

`virtual void` **putPacket(** `const PacketHeader& pktHeader, const char *pktData );`

Write the packet header and data (if it exists) to the pipe

Name

CString – Character string manipulation

Authors

Daniel A. Reed (reed@cs.uiuc.edu)

Description

A CString supports the "standard" string operations, including creation, concatenation (+), comparison (<, <=, >, >=, !=, ==), and assignment (=).

Friends

Global method operator<.
Global method operator>.
Global method operator<=.
Global method operator>=.
Global method operator==.
Global method operator!=.
Global method operator+.
Global method concat.

Available Methods

Public

- char* – Conversion to char*
- concat – CString and character string concatenation
- concat – CString concatenation
- getValue – Extract C string representation
- length – Character string length
- operator!= – Lexiographic CString inequality test
- operator!= – Lexiographic char inequality test
- operator+ – CString and character string concatenation
- operator+ – CString concatenation
- operator<= – Lexiographic CString comparison
- operator= – CString to CString assignment
- operator= – Character string to CString assignment
- operator== – Lexiographic CString equality test
- operator== – Lexiographic char equality test
- operator>= – Lexiographic character string/CString "greater than or equal to"

- operator[] – CString character extraction
- printOn – helper function for CString output

Included Files

"Obj.h"

Public Methods

CString( );

The simplest CString constructor

CString(`const char *stringVal`);

Initialized CString/character string constructor

CString(`CString& str`);

Initialized CString/CString constructor

CString(`const CString& str`);

Initialized CString/CString constructor

`virtual` ~**CString(**);

The CString destructor

`CString` **concat(** `const CString& str`);

CString concatenation

`CString` **concat(** `const char *charstr`);

CString and character string concatenation

`const char *` **getValue(**) `const;`

Extract C string representation

`int` **length(**) `const;`

Character string length

`CString&` **operator=(** `const char *charstr`);

Character string to CString assignment

```
CString& operator=(   const CString& str );
```
CString to CString assignment

```
int operator==(   const CString& str ) const;
```
Lexiographic CString equality test

```
int operator==(   const char *str ) const;
```
Lexiographic char equality test

```
int operator<=(   const CString& str ) const;
```
Lexiographic CString comparison

```
int operator>=(   const CString& str ) const;
```
Lexiographic character string/CString "greater than or equal to"

```
int operator!=(   const CString& str ) const;
```
Lexiographic CString inequality test

```
int operator!=(   const char *str ) const;
```
Lexiographic char inequality test

```
CString operator+(   const CString& str );
```
CString concatenation

```
CString operator+(   const char *charstr );
```
CString and character string concatenation

```
char& operator[](   int loc ) const;
```
CString character extraction

```
operator const char*(   ) const;
```
Conversion to char*

```
virtual void printOn(   ostream& strm = cout ) const;
```
helper function for CString output

Name

ConversionPipeReader – Handles input from a pipe connected to a binary input file whose data characteristics do not match those of the native machine.

Authors

Ruth A. Aydt (aydt@cs.uiuc.edu)

Heritage

Inherits from – PipeReader, Obj.

Description

The ConversionPipeReader class implements methods which handle reading input from a binary file whose data characteristics do not match those of the native machine. This is accomplished primarily by calling the appropriate parser methods in the classes corresponding to the sort of data we are reading (Descriptors, Attributes, ...).

Available Methods

Private

- _readPacket – Grabs a packet from the pipe, updating currentHeader and pipeAttributes as appropriate

Protected

- _setClassName (inherited) – Set _className

Public

- abort (inherited) – Print message and abort system.
- error (inherited) – This prints an error message.
- gentleWarning (inherited) – A warning message to stderr
- getAttributes (inherited) – Copy pipeAttributes to attributes
- getClassName (inherited) – Return pointer to class name
- getData – Parse the record data in interal buffer into recordDossier.
- getDescriptor – Parse the structure descriptor in internal buffer into structDescr
- getFileSize (inherited) – Returns the size of the file the pipe is connected to. Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- getObjectName (inherited) – This returns a name for the object. As defined in Obj, it returns the same string as the is returned by getClassName. However, derived classes may redefine it to include more detailed information.
- getOffset (inherited) – Returns the current offset into the pipe being read from. Not valid for all types of pipes - if call made and invalid, program aborts with error message.

- getPacketDataPtr – Virtual method that does not make sense for this class & is not implemented.
- getPacketHeader – Get header of packet next in pipe. If packet is there, it is actually read into packetBuffer
- hash (inherited) – Return a hash value for this object
- info (inherited) – Prints informational message.
- inspect (inherited) – Inspect the object.
- notImplemented (inherited) – Print an error message that the argument list is not implemented
- printOn (inherited) – Helper function for output
- rewind (inherited) – Rewind the offset to what it was when the PipeReader was created. Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- setOffset (inherited) – Sets the current offset into the pipe being read from. Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- shouldNotImplement (inherited) – This method prints an error message that says it should not be implemented.
- subclassResponsibility (inherited) – Method prints an error message that says it should be implemented by a subclass.
- successfulConnection (inherited) – Returns TRUE_ if the connection was successful in the constructor.
- warning (inherited) – This prints a warning message.

Included Files

"Defines.h"
"PipeReader.h"

Private Methods

`Boolean_ _readPacket( );`

Grabs a packet from the pipe, updating currentHeader and pipeAttributes as appropriate

Public Methods

ConversionPipeReader(`StreamPipe *initPipe );`

The constructor

~ConversionPipeReader(`);`

`virtual Boolean_ getData( RecordDossier& recordDossier );`

Parse the record data in interal buffer into recordDossier.

`virtual Boolean_` **getDescriptor(** `StructureDescriptor& structDescr );`

Parse the structure descriptor in internal buffer into structDescr

`virtual const char *` **getPacketDataPtr(** `) const ;`

Virtual method that does not make sense for this class & is not implemented.

`PacketHeader` **getPacketHeader(** `);`

Get header of packet next in pipe. If packet is there, it is actually read into packetBuffer

Name

ConversionPipeWriter – Handles output to a pipe connected to a binary output file whose data characteristics do not match those of the native machine.

Authors

Ruth A. Aydt (aydt@cs.uiuc.edu)

Heritage

Inherits from – PipeWriter, Obj.

Description

ConversionPipeWriter is the class which writes binary SDDF data with characteristics different than those of the native machine to a pipe connected to an output file.

Available Methods

Private

- _writePacket – Write the packet constructed in packetBuffer to the pipe.

Protected

- _setClassName (inherited) – Set _className
- _writeSddfHeader – Write the SDDF header information to the current Stream Pipe.

Public

- abort (inherited) – Print message and abort system.
- addPipe – Add another pipe to the list we write to.
- deletePipe (inherited) – Delete the pipe from the list we write to.
- error (inherited) – This prints an error message.
- gentleWarning (inherited) – A warning message to stderr
- getClassName (inherited) – Return pointer to class name
- getFileSize (inherited) – Returns the size of the file the pipe is connected to. Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- getObjectName (inherited) – This returns a name for the object. As defined in Obj, it returns the same string as the is returned by getClassName. However, derived classes may redefine it to include more detailed information.
- getOffset (inherited) – Returns the current offset into the pipe being written to. Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- hash (inherited) – Return a hash value for this object
- info (inherited) – Prints informational message.
- inspect (inherited) – Inspect the object.
- notImplemented (inherited) – Print an error message that the argument list is not implemented

- printOn (inherited) – Stream output function
- putAttributes – Update pipeAttributes and write them to the pipe.
- putCommand – Write the command to the pipe
- putData – Write the data associated with a record to the pipe.
- putDescriptor – Write the Structure Descriptor to the pipe.
- putPacket – Virtual method that does not make sense for this class and is not implemented.
- rewind (inherited) – Rewind the offset to what it was when the PipeWriter was created and the header written (file is truncated to the shorter length). Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- setOffset (inherited) – Sets the current offset into the pipe being read from (no change in data takes place). Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- shouldNotImplement (inherited) – This method prints an error message that says it should not be implemented.
- subclassResponsibility (inherited) – Method prints an error message that says it should be implemented by a subclass.
- successfulConnection (inherited) – Returns TRUE_ if the connection was successful in the constructor.
- warning (inherited) – This prints a warning message.

Included Files

"DataCharacteristics.h"
"PipeWriter.h"

Private Methods

`void` **_writePacket**`(  );`

Write the packet constructed in packetBuffer to the pipe.

Protected Methods

`virtual void` **_writeSddfHeader**`(  ) const;`

Write the SDDF header information to the current Stream Pipe.

Public Methods

ConversionPipeWriter`(  StreamPipe *thePipe );`

A constructor with no specification of the output file's characteristics. The file will be written with the same characteristics as the native machine EXCEPT the byte order will be reversed.

ConversionPipeWriter(`StreamPipe *thePipe,`
`const DataCharacteristics& outCharacteristics );`

A constructor with explicit specification of the output file's data characteristics.

~ConversionPipeWriter( );

The destructor

`virtual Boolean_` **addPipe(** `StreamPipe *thePipe );`

Add another pipe to the list we write to.

`virtual void` **putAttributes(**);

Write pipeAttributes to the pipe

`virtual void` **putAttributes(** `const Attributes& attributes );`

Update pipeAttributes and write them to the pipe.

`virtual void` **putCommand(** `int pktTag = 0 );`

Write the command to the pipe

`virtual void` **putData(** `const RecordDossier& recordDossier );`

Write the data associated with a record to the pipe.

`virtual void` **putDescriptor(** `const StructureDescriptor& structDescr, int pktTag );`

Write the Structure Descriptor to the pipe.

`virtual void` **putPacket(** `const PacketHeader&, const char * ) ;`

Virtual method that does not make sense for this class and is not implemented.

Name

DataCharacteristics – Class used to define the characteristics of data represented in Binary SDDF format. Such things as byte ordering and size of different data types are encapsulated here, along with methods to manipulate between native machine representation and a non-native format.

Authors

Ruth A. Aydt (aydt@cs.uiuc.edu)
Contributing Author: Bill Whitehouse (whitehou@cs.uiuc.edu)

Heritage

Inherits from – Obj.

Description

The DataCharacteristics Class knows about the various representations of data types for different machines. It reads and writes headers to binary SDDF files and is used to convert between other representations and the representations on the native machine.

Available Methods

Protected

- _setClassName (inherited) – Set _className

Public

- abort (inherited) – Print message and abort system.
- error (inherited) – This prints an error message.
- functionGet – Gets appropriate number of bytes from **bufptr into char, advances *bufptr by the number of bytes read, and returns the number of bytes read.
- functionPut – Writes the char value as bytes in **bufptr, advances *bufptr by the number of bytes written and returns the number of bytes written
- gentleWarning (inherited) – A warning message to stderr
- getAtomSize – Returns the size of the given MachineDataType on the non-native machine.
- getClassName (inherited) – Return pointer to class name
- getObjectName (inherited) – This returns a name for the object. As defined in Obj, it returns the same string as the is returned by getClassName. However, derived classes may redefine it to include more detailed information.
- hash (inherited) – Return a hash value for this object
- info (inherited) – Prints informational message.
- initializeFromFile – Reads the header information from the input file and sets the class variables using that information. Returns FALSE_ if for some reason it's unable to parse the input or if the needed conversion routines don't exist.

- inspect (inherited) – Inspect the object.
- intSize – Returns the size of an integer
- notImplemented (inherited) – Print an error message that the argument list is not implemented
- notNative – Returns TRUE_ if the representations don't match those of the native machine
- printOn (inherited) – Stream output function
- setByteOrdering – Sets byte ordering
- setByteOrderingNonNative – Sets byte ordering to be the opposite of what it is on the native machine.
- setBytesInChar – Sets number of bytes in char
- setBytesInDouble – Sets number of bytes in double
- setBytesInFloat – Sets number of bytes in float
- setBytesInInteger – Sets number of bytes in int
- setBytesInLong – Sets number of bytes in long (Currently this type is not supported in SDDF)
- setBytesInLongDouble – Sets number of bytes in long double (Currently this type is not supported in SDDF)
- setBytesInShort – Sets number of bytes in short (Currently this type is not supported in SDDF)
- setCharRepresentation – Sets character representation
- setDoubleFloatRepresentation – Sets double representation
- setExtraFloatRepresentation – Sets long double representation (Currently this extra precision floating point is not supported in SDDF)
- setIntegerRepresentation – Sets integer representation
- setSingleFloatRepresentation – Sets float representation
- shouldNotImplement (inherited) – This method prints an error message that says it should not be implemented.
- subclassResponsibility (inherited) – Method prints an error message that says it should be implemented by a subclass.
- warning (inherited) – This prints a warning message.
- writeHeader – Writes a SDDF binary header with the characteristics for the instance of the class into the buffer. Returns the number of bytes written or -1 if an error occured.
- writeNativeHeader – Writes a SDDF binary header with the representations native to the machine where the program is being executed into the buffer. Returns the number of bytes written or -1 if an error occured.

Included Files

"Defines.h"
"DataTraits.h"
"Obj.h"

Public Methods

DataCharacteristics( );

Constructor

`static int` **writeNativeHeader**(`char *const bufPtr, int bufLen`);

Writes a SDDF binary header with the representations native to the machine where the program is being executed into the buffer. Returns the number of bytes written or -1 if an error occured.

`int` **functionGet**(`const char *buffer, char *value ) const;`

Gets appropriate number of bytes from *buffer into char and returns the number of bytes read.

`int` **functionGet**(`char **bufptr, char *value ) const;`

Gets appropriate number of bytes from **bufptr into char, advances *bufptr by the number of bytes read, and returns the number of bytes read.

`int` **functionGet**(`const char *buffer, int *value ) const;`

Gets appropriate number of bytes from *buffer into int and returns the number of bytes read.

`int` **functionGet**(`char **bufptr, int *value ) const;`

Gets appropriate number of bytes from **bufptr into int, advances *bufptr by the number of bytes read, and returns the number of bytes read.

`int` **functionGet**(`const char *buffer, double *value ) const;`

Gets appropriate number of bytes from *buffer into double and returns the number of bytes read.

`int` **functionGet**(`char **bufptr, double *value ) const;`

Gets appropriate number of bytes from **bufptr into double, advances *bufptr by the number of bytes read, and returns the number of bytes read.

`int` **functionGet**(`const char *buffer, float *value ) const;`

Gets appropriate number of bytes from *buffer into float and returns the number of bytes read.

`int` **functionGet**(`char **bufptr, float *value ) const;`

Gets appropriate number of bytes from **bufptr into float, advances *bufptr by the number of bytes read, and returns the number of bytes read.

`int` **functionPut**(`char *buffer, const char *value ) const;`

Writes the char value as bytes in *buffer and returns the number of bytes written

`int` **functionPut(** `char **bufptr, const char *value ) const;`

Writes the char value as bytes in **bufptr, advances *bufptr by the number of bytes written and returns the number of bytes written

`int` **functionPut(** `char *buffer, const int *value ) const;`

Writes the int value as bytes in *buffer and returns the number of bytes written

`int` **functionPut(** `char **bufptr, const int *value ) const;`

Writes the int value as bytes in **bufptr, advances *bufptr by the number of bytes written and returns the number of bytes written

`int` **functionPut(** `char *buffer, const double *value ) const;`

Writes the double value as bytes in *buffer and returns the number of bytes written

`int` **functionPut(** `char **bufptr, const double *value ) const;`

Writes the double value as bytes in **bufptr, advances *bufptr by the number of bytes written and returns the number of bytes written

`int` **functionPut(** `char *buffer, const float *value ) const;`

Writes the float value as bytes in *buffer and returns the number of bytes written

`int` **functionPut(** `char **bufptr, const float *value ) const;`

Writes the float value as bytes in **bufptr, advances *bufptr by the number of bytes written and returns the number of bytes written

`int` **getAtomSize(** `MachineDataType atomType ) const;`

Returns the size of the given MachineDataType on the non-native machine.

`Boolean_` **initializeFromFile(** `InputFileStreamPipe * pipe, int fileDescr );`

Reads the header information from the input file and sets the class variables using that information. Returns FALSE_ if for some reason it's unable to parse the input or if the needed conversion routines don't exist.

`int` **intSize(** `) const;`

Returns the size of an integer

`Boolean_` **notNative(** `) const;`

Returns TRUE_ if the representations don't match those of the native machine

`Boolean_` **setByteOrdering(** `Sddf_BYTE_ORDERING newByteOrdering );`

Sets byte ordering

`Boolean_` **setByteOrderingNonNative**`( );`

Sets byte ordering to be the opposite of what it is on the native machine.

`Boolean_` **setIntegerRepresentation**`( Sddf_INT_REPRESENT newIntegerRepresentation );`

Sets integer representation

`Boolean_` **setSingleFloatRepresentation**`( Sddf_SNGLP_REPRESENT newSingleFloatRepresentation );`

Sets float representation

`Boolean_` **setDoubleFloatRepresentation**`( Sddf_DBLP_REPRESENT newDoubleFloatRepresentation );`

Sets double representation

`Boolean_` **setExtraFloatRepresentation**`( Sddf_EXTRA_REPRESENT newExtraFloatRepresentation );`

Sets long double representation (Currently this extra precision floating point is not supported in SDDF)

`Boolean_` **setCharRepresentation**`( Sddf_CHAR_REPRESENT newCharRepresentation );`

Sets character representation

`Boolean_` **setBytesInChar**`( int newBytesInChar );`

Sets number of bytes in char

`Boolean_` **setBytesInShort**`( int newBytesInShort );`

Sets number of bytes in short (Currently this type is not supported in SDDF)

`Boolean_` **setBytesInInteger**`( int newBytesInInteger );`

Sets number of bytes in int

`Boolean_` **setBytesInLong**`( int newBytesInLong );`

Sets number of bytes in long (Currently this type is not supported in SDDF)

`Boolean_` **setBytesInFloat**`( int newBytesInFloat );`

Sets number of bytes in float

`Boolean_` **setBytesInDouble**`( int newBytesInDouble );`

Sets number of bytes in double

`Boolean_` **setBytesInLongDouble**`( int newBytesInLongDouble );`

Sets number of bytes in long double (Currently this type is not supported in SDDF)

`int` **writeHeader(** `char *const bufPtr, int bufLen ) const;`

Writes a SDDF binary header with the characteristics for the instance of the class into the buffer. Returns the number of bytes written or -1 if an error occured.

Name

DataTraits – Header file for class used to define the characteristics of data in the Pablo system. Contains information on the base MachineDataType and the dimesions of the data.

Authors

Ruth A. Aydt (aydt@cs.uiuc.edu)

Description

The DataTraits Class provides an interface to type and dimesion information about the data.

Available Methods

Public

- clear – Set type to UNDEFINED and dimension to zero.
- getAtomSize – Returns sizeof() atom of this type
- getDimension – Returns the dimension
- getType – Returns the type
- getTypeName – Returns the name of the type
- isArray – Returns TRUE_ if dimension $>$ 0, otherwise returns FALSE_
- isDefined – Returns TRUE_ if type is not UNDEFINED
- isScalar – Returns TRUE_ if dimension $==$ 0, otherwise returns FALSE_
- isUndefined – Returns TRUE_ if type is UNDEFINED
- loadConfigurationFromFP – Load info from file pointer
- operator!= – Inequality operator
- operator= – Assignment operator
- operator== – Equality operator
- printOn – Helper function for output
- saveConfigurationToFP – Save info to file pointer
- setDimension – Set the dimension
- setTraits – Set both dimension and type
- setType – Set the type

Included Files

<stdio.h>
<stream.h>
"Defines.h"

Public Methods

`static DataTraits * loadConfigurationFromFP( FILE *fp );`

Load info from file pointer

`DataTraits( MachineDataType _type = UNDEFINED, int _dimension = 0 );`

Constructor with type and dimension specified (with defaults of type = UNDEFINED and dimension = 0)

`DataTraits( const DataTraits& _traits );`

Constructor with copy initialization

`void clear( );`

Set type to UNDEFINED and dimension to zero.

`int getAtomSize( ) const;`

Returns sizeof() atom of this type

`int getDimension( ) const;`

Returns the dimension

`MachineDataType getType( ) const;`

Returns the type

`const char * getTypeName( ) const;`

Returns the name of the type

`Boolean_ isArray( ) const;`

Returns TRUE_ if dimension > 0, otherwise returns FALSE_

`Boolean_ isDefined( ) const;`

Returns TRUE_ if type is not UNDEFINED

`Boolean_ isScalar( ) const;`

Returns TRUE_ if dimension $== 0$, otherwise returns FALSE_

`Boolean_ isUndefined( ) const;`

Returns TRUE_ if type is UNDEFINED

`Boolean_` **saveConfigurationToFP**`( FILE *fp ) const;`

Save info to file pointer

`void` **setDimension**`( int _dimension );`

Set the dimension

`void` **setTraits**`( MachineDataType _type, int _dimension );`

Set both dimension and type

`void` **setType**`( MachineDataType _type );`

Set the type

`DataTraits&` **operator**`=( const DataTraits& _traits );`

Assignment operator

`Boolean_` **operator**`==( const DataTraits& _traits ) const;`

Equality operator

`Boolean_` **operator**`!=( const DataTraits& _traits ) const;`

Inequality operator

`void` **printOn**`( ostream& strm ) const;`

Helper function for output

Name

FieldDescriptor – Basic class used to describe a field of data in the self-documenting data format.

Authors

Ruth A. Aydt (aydt.cs.uiuc.edu)
Contributing Author: Daniel A. Reed (reed@cs.uiuc.edu)

Heritage

Inherits from – BaseDescriptor, Obj.

Description

A FieldDescriptor is the base component of a self documenting data format. It defines a one field of data.

Available Methods

Protected

- _setClassName (inherited) – Set _className

Public

- abort (inherited) – Print message and abort system.
- bitsToObj – Method used to translate from binary bitstream representation of data to FieldDescriptor instance. Returns number of bytes read or -1 if some error occured.
- bytesNeeded – Returns the number of bytes needed to represent the instance as a binary bitstream
- cbitsToObj – Method used to translate from non-native binary bitstream representation of data to FieldDescriptor instance. Returns number of bytes read or -1 if some error occured.
- cbytesNeeded – Returns the number of bytes needed to represent the instance as a non-native binary bitstream
- error (inherited) – This prints an error message.
- gentleWarning (inherited) – A warning message to stderr
- getAttributes (inherited) – Returns the attribute dictionary for this component
- getClassName (inherited) – Return pointer to class name
- getDimension – Returns the dimension of the field (zero for scalars)
- getName (inherited) – Returns the name of the component
- getObjectName (inherited) – Overloads Obj::getObjectName() to return the full name including class and component names.
- getTraits – Returns the DataTraits for this field.
- getType – Returns the builtin data type of this field (e.g., int)
- hash (inherited) – Return a hash value for this object

- info (inherited) – Prints informational message.
- insertAttributes (inherited) – Insert a new attribute
- inspect (inherited) – Inspect the object.
- notImplemented (inherited) – Print an error message that the argument list is not implemented
- notNull (inherited) – Returns FALSE_ if name is NOMATCH, otherwise returns TRUE_.
- objToBits – Used to translate instance to binary bitstream representation of data. Returns number of bytes used to write bitstream data, or -1 if bufLen < number of bytes needed.
- objToCbits – Used to translate instance to non-native binary bitstream representation of data. Returns number of bytes used to write bitstream data, or -1 if bufLen < number of bytes needed.
- operator!= – FieldDescriptor non-equivalence
- operator= – FieldDescriptor assigment
- operator== – FieldDescriptor equivalence
- printOn – Helper function for FieldDescriptor output.
- setAttributes (inherited) – Set the attribute dictionary for this component
- setDimension – Set the dimension of the field (zero for scalars)
- setName (inherited) – Set the name of the component. Also update objectName;
- setTraits – Set the DataTraits for field
- setType – Set the builtin data type of this field (e.g., int)
- shouldNotImplement (inherited) – This method prints an error message that says it should not be implemented.
- subclassResponsibility (inherited) – Method prints an error message that says it should be implemented by a subclass.
- warning (inherited) – This prints a warning message.

Included Files

"BaseDescriptor.h"
"DataTraits.h"
"Defines.h"

Public Methods

FieldDescriptor( );

A FieldDescriptor constructor with no initialization

FieldDescriptor(const CString& _name);

A FieldDescriptor constructor with initial value for name

```
FieldDescriptor( const CString& _name,  const Attributes& _attributes,
                 const MachineDataType _type,  const int _dimension );
```

A FieldDescriptor constructor with initial values for all elements

FieldDescriptor(`const FieldDescriptor& field );`

FieldDescriptor constructor with copy initialization

`int` **bitsToObj(** `const char *bufPtr );`

Method used to translate from binary bitstream representation of data to FieldDescriptor instance. Returns number of bytes read or -1 if some error occured.

`int` **bytesNeeded(** `) const;`

Returns the number of bytes needed to represent the instance as a binary bitstream

`int` **cbitsToObj(** `const char *bufPtr, const DataCharacteristics *cnvInfo );`

Method used to translate from non-native binary bitstream representation of data to FieldDescriptor instance. Returns number of bytes read or -1 if some error occured.

`int` **cbytesNeeded(** `const DataCharacteristics *cnvInfo ) const;`

Returns the number of bytes needed to represent the instance as a non-native binary bitstream

`int` **getDimension(** `) const;`

Returns the dimension of the field (zero for scalars)

`const DataTraits&` **getTraits(** `) const;`

Returns the DataTraits for this field.

`MachineDataType` **getType(** `) const;`

Returns the builtin data type of this field (e.g., int)

`int` **objToBits(** `char* const bufPtr, int bufLen ) const;`

Used to translate instance to binary bitstream representation of data. Returns number of bytes used to write bitstream data, or -1 if bufLen $<$ number of bytes needed.

`int` **objToCbits(** `char* const bufPtr, int bufLen,`
`const DataCharacteristics *cnvInfo ) const;`

Used to translate instance to non-native binary bitstream representation of data. Returns number of bytes used to write bitstream data, or -1 if bufLen $<$ number of bytes needed.

`void` **setDimension(** `const int fieldDimension );`

Set the dimension of the field (zero for scalars)

`void setTraits( const DataTraits& _traits );`

Set the DataTraits for field

`void setType( const MachineDataType _type );`

Set the builtin data type of this field (e.g., int)

`FieldDescriptor& operator=( const FieldDescriptor& field );`

FieldDescriptor assigment

`Boolean_ operator==( const FieldDescriptor& field ) const;`

FieldDescriptor equivalence

`Boolean_ operator!=( const FieldDescriptor& field ) const;`

FieldDescriptor non-equivalence

`virtual void printOn( ostream& strm = cout ) const;`

Helper function for FieldDescriptor output.

Name

FileStreamPipe – Base class for file-based pipes.

Authors

Robert Olson (olson@cs.uiuc.edu)
Contributing Author: Ruth A. Aydt (aydt@cs.uiuc.edu)

Heritage

Inherits from – StreamPipe, BasePipe, Obj.

Description

A FileStreamPipe is a base class for file-based pipes.

Available Methods

Protected
- _setClassName (inherited) – Set _className
- _setObjectName (inherited) – Give detailed name to Pipe

Public
- abort (inherited) – Print message and abort system.
- bytesFree (inherited) – Return the number of bytes that can be written
- bytesReady (inherited) – Return the number of bytes ready to be read
- createPipeReader (inherited) – Implemented by derived classes to create appropriate type of PipeReader for the StreamPipe. Returns NULL if can't create PipeReader.
- error (inherited) – This prints an error message.
- gentleWarning (inherited) – A warning message to stderr
- get (inherited) – Get data from the pipe
- getByteOffset (inherited) – Returns the current byte offset in the pipe. This is not valid for all types of pipes. Implemented in appropriate derived classes.
- getClassName (inherited) – Return pointer to class name
- getFileDescriptor – Return the file descriptor of the file associated with the pipe.
- getFileSize (inherited) – Returns the length of the file. Does not change the current file pointer. This is not valid for all types of pipes. Implemented in appropriate derived classes.
- getInputCharacteristics (inherited) – Return pointer to the data characteristics for the input associated with this pipe. Only valid if inputType is CONVERT_BINARY_SDDF. Implemented in appropriate derived classes.
- getInputType (inherited) – Return the type of the input associated with this pipe. Returns valid type in appropriate derived classes.
- getInputWrapper (inherited) – Get the input wrapper

- getObjectName (inherited) – Overloads Obj::getObjectName() to return _objectName
- getOutputWrapper (inherited) – Get the output wrapper
- getSkippingPeek (inherited) – Get data from the pipe, skipping over data in the peek buffer. Also clear peek buffer.
- hash (inherited) – Return a hash value for this object
- info (inherited) – Prints informational message.
- inspect (inherited) – Inspect the object.
- isEmpty (inherited) – Returns TRUE_ if the pipe is empty
- isFilePipe – Always returns TRUE_
- isFull (inherited) – Returns TRUE_ if the pipe is full
- isInputPipe (inherited) – Returns TRUE_ if this is an input pipe
- isOutputPipe (inherited) – Returns TRUE_ if this is an output pipe
- notImplemented (inherited) – Print an error message that the argument list is not implemented
- peek (inherited) – Peek at data in the pipe
- printOn – Stream output function
- put (inherited) – Write data into the pipe
- setByteOffset (inherited) – Sets the current byte offset in the pipe. This is not valid for all types of pipes. Implemented in appropriate derived classes.
- setInputWrapper (inherited) – Set the input wrapper
- setOutputWrapper (inherited) – Set the output wrapper
- shouldNotImplement (inherited) – This method prints an error message that says it should not be implemented.
- subclassResponsibility (inherited) – Method prints an error message that says it should be implemented by a subclass.
- successfulOpen – Returns true if there is a file associated with the pipe.
- truncateAfterByte (inherited) – Truncates the data in the pipe after the specified byte. This is only valid for pipes associated with output data files.
- warning (inherited) – This prints a warning message.

Included Files

"StreamPipe.h"

Public Methods

FileStreamPipe( );

The FileStreamPipe constructor

~FileStreamPipe( );

The FileStreamPipe destructor

`virtual Boolean_` **isFilePipe**`( ) const ;`

Always returns TRUE_

`int` **getFileDescriptor**`( ) const;`

Return the file descriptor of the file associated with the pipe.

`Boolean_` **successfulOpen**`( ) const;`

Returns true if there is a file associated with the pipe.

`virtual void` **printOn**`( ostream& os ) const;`

Stream output function

Name

InputFileStreamPipe – A StreamPipe that reads from a file.

Authors

Robert Olson (olson@cs.uiuc.edu)
Contributing Author: Ruth Aydt (aydt@cs.uiuc.edu)
Contributing Author: Bill Whitehouse (whitehou@cs.uiuc.edu)

Heritage

Inherits from – FileStreamPipe, StreamPipe, BasePipe, Obj.

Description

An InputFileStreamPipe is a pipe that takes its input from a disk file.

Available Methods

Private
- _readPool – Reads from beginning of the pre-fetch pool

Protected
- _setClassName (inherited) – Set _className
- _setObjectName (inherited) – Give detailed name to Pipe

Public
- abort (inherited) – Print message and abort system.
- bytesFree – Returns the number of bytes that can be written
- bytesReady – Returns the number of bytes ready to read
- createPipeReader – Creates the appropriate type of Pipe Reader for the input file we're processing and return it to the caller. Returns NULL if unable to create the reader for some reason.
- error (inherited) – This prints an error message.
- gentleWarning (inherited) – A warning message to stderr
- get – Read data from the file, returning number of bytes read.
- getByteOffset – Returns the current byte offset in the file associated with this pipe. Note that calls to peek() do not advance the offset.
- getClassName (inherited) – Return pointer to class name
- getFileDescriptor (inherited) – Return the file descriptor of the file associated with the pipe.
- getFileSize – Returns the length of the file. Does not change the current offset.
- getInputCharacteristics – Returns pointer to the information on the data characteristics for the input file. This will be NULL if the file isn't in Binary SDDF format.
- getInputType – Returns SDDF type of the input file.

- getInputWrapper (inherited) – Get the input wrapper
- getObjectName (inherited) – Overloads Obj::getObjectName() to return _objectName
- getOutputWrapper (inherited) – Get the output wrapper
- getSkippingPeek – Read data from the file, skipping any data that is in the peek buffer. Also clears peek buffer.
- hash (inherited) – Return a hash value for this object
- info (inherited) – Prints informational message.
- inspect (inherited) – Inspect the object.
- isEmpty – Returns TRUE_ if the pipe is empty
- isFilePipe (inherited) – Always returns TRUE_
- isFull – Returns TRUE_ if the pipe is full
- isInputPipe – Always returns TRUE_
- isOutputPipe (inherited) – Returns TRUE_ if this is an output pipe
- notImplemented (inherited) – Print an error message that the argument list is not implemented
- peek – Peek from the file
- printOn – Stream output function
- put – It is an error to put to a read-file
- setByteOffset – Sets the current byte offset in the file associated with this pipe. Any bytes in the peekBuffer at the time this call is made are cleared.
- setInputWrapper (inherited) – Set the input wrapper
- setOutputWrapper (inherited) – Set the output wrapper
- shouldNotImplement (inherited) – This method prints an error message that says it should not be implemented.
- subclassResponsibility (inherited) – Method prints an error message that says it should be implemented by a subclass.
- successfulOpen (inherited) – Returns true if there is a file associated with the pipe.
- truncateAfterByte (inherited) – Truncates the data in the pipe after the specified byte. This is only valid for pipes associated with output data files.
- warning (inherited) – This prints a warning message.

Included Files

"FileStreamPipe.h"

Private Methods

```
int _readPool(   char* theBuffer,  int amountToRead );
```

Reads from beginning of the pre-fetch pool

Public Methods

```
InputFileStreamPipe(   const char *fileName,
                       int bytesInReadPool = DEFAULTreadPoolSize );
```

The constructor

~InputFileStreamPipe( );

The destructor

`virtual int` **bytesFree**( ) `const;`

Returns the number of bytes that can be written

`virtual int` **bytesReady**( ) `const;`

Returns the number of bytes ready to read

`virtual PipeReader *` **createPipeReader**( ) `const;`

Creates the appropriate type of Pipe Reader for the input file we're processing and return it to the caller. Returns NULL if unable to create the reader for some reason.

`virtual int` **get**(`char *data, int lengthRequested`);

Read data from the file, returning number of bytes read.

`virtual const DataCharacteristics *` **getInputCharacteristics**( ) `const;`

Returns pointer to the information on the data characteristics for the input file. This will be NULL if the file isn't in Binary SDDF format.

`virtual off_t` **getByteOffset**( );

Returns the current byte offset in the file associated with this pipe. Note that calls to peek() do not advance the offset.

`virtual off_t` **getFileSize**( );

Returns the length of the file. Does not change the current offset.

`virtual PabloSddfType` **getInputType**( ) `const;`

Returns SDDF type of the input file.

`virtual int` **getSkippingPeek**(`char *data, int lengthRequested`);

Read data from the file, skipping any data that is in the peek buffer. Also clears peek buffer.

`virtual Boolean_` **isEmpty**( ) `const;`

Returns TRUE_ if the pipe is empty

`virtual Boolean_` **isFull**( ) `const;`

Returns TRUE_ if the pipe is full

`virtual Boolean_ isInputPipe( ) const ;`

Always returns TRUE_

`virtual int peek( char *data, int lengthRequested );`

Peek from the file

`virtual int put( const char *, int );`

It is an error to put to a read-file

`virtual void setByteOffset( off_t newByteOffset );`

Sets the current byte offset in the file associated with this pipe. Any bytes in the peekBuffer at the time this call is made are cleared.

`virtual void printOn( ostream& os ) const;`

Stream output function

Name

Obj – Base class for all other derived classes

Authors

Daniel A. Reed (reed@cs.uiuc.edu)
Contributing Author: Ruth A. Aydt (aydt@cs.uiuc.edu)

Description

Obj provides support for standard error and warning messages, and maintains the name of the object.

Available Methods

Protected
- _setClassName – Set _className

Public
- abort – Print message and abort system.
- error – This prints an error message.
- gentleWarning – A warning message to stderr
- getClassName – Return pointer to class name
- getObjectName – This returns a name for the object. As defined in Obj, it returns the same string as the is returned by getClassName. However, derived classes may redefine it to include more detailed information.
- hash – Return a hash value for this object
- info – Prints informational message.
- inspect – Inspect the object.
- notImplemented – Print an error message that the argument list is not implemented
- printOn – Stream output function
- shouldNotImplement – This method prints an error message that says it should not be implemented.
- subclassResponsibility – Method prints an error message that says it should be implemented by a subclass.
- warning – This prints a warning message.

Included Files

<stream.h>
"Assert.h"
"Defines.h"

Protected Methods

`void` **_setClassName**`( const char *cname );`

Set _className

Public Methods

Obj`( );`

The Obj constructor

Obj`( const char *cname );`

Obj constructor with class name

`virtual ~`**Obj**`( );`

The Obj destructor

`virtual void` **abort**`( const char *s, ... ) const;`

Print message and abort system.

`virtual void` **error**`( const char *s, ... ) const;`

This prints an error message.

`virtual void` **gentleWarning**`( const char *s, ... ) const;`

A warning message to stderr

`const char *` **getClassName**`( ) const;`

Return pointer to class name

`virtual const char *` **getObjectName**`( ) const;`

This returns a name for the object. As defined in Obj, it returns the same string as the is returned by getClassName. However, derived classes may redefine it to include more detailed information.

`virtual void` **info**`( const char *s, ... ) const;`

Prints informational message.

`virtual void` **inspect**`( ) const;`

Inspect the object.

`virtual void` **warning**`( const char *s, ... ) const;`

This prints a warning message.

`int` **hash**`( ) const;`

Return a hash value for this object

`void` **notImplemented**`( const char *s, ... ) const;`

Print an error message that the argument list is not implemented

`void` **shouldNotImplement**`( const char *s, ... ) const;`

This method prints an error message that says it should not be implemented.

`void` **subclassResponsibility**`( const char *s, ... ) const;`

Method prints an error message that says it should be implemented by a subclass.

`virtual void` **printOn**`( ostream &os = cout ) const ;`

Stream output function

Name

OutputFileStreamPipe – A StreamPipe that writes to a file.

Authors

Robert Olson (olson@cs.uiuc.edu)
Contributing Author: Ruth Aydt (aydt@cs.uicu.edu)
Contributing Author: Bill Whitehouse (whitehou@cs.uiuc.edu)

Heritage

Inherits from – FileStreamPipe, StreamPipe, BasePipe, Obj.

Description

An OutputFileStreamPipe is a pipe that writes it output to a disk file.

Available Methods

Private
- _flushPool – Flushes data in the buffer pool to the output file.

Protected
- _setClassName (inherited) – Set _className
- _setObjectName (inherited) – Give detailed name to Pipe

Public
- abort (inherited) – Print message and abort system.
- bytesFree – Returns the number of bytes that can be written
- bytesReady – Returns the number of bytes ready to read
- createPipeReader – Not implemented for class
- error (inherited) – This prints an error message.
- gentleWarning (inherited) – A warning message to stderr
- get – It is an error to read from an output file
- getByteOffset – Returns the current byte offset in the file associated with this pipe.
- getClassName (inherited) – Return pointer to class name
- getFileDescriptor (inherited) – Return the file descriptor of the file associated with the pipe.
- getFileSize – Returns the length of the file. Does not change the current file pointer.
- getInputCharacteristics (inherited) – Return pointer to the data characteristics for the input associated with this pipe. Only valid if inputType is CONVERT_BINARY_SDDF. Implemented in appropriate derived classes.
- getInputType (inherited) – Return the type of the input associated with this pipe. Returns valid type in appropriate derived classes.
- getInputWrapper (inherited) – Get the input wrapper

- getObjectName (inherited) – Overloads Obj::getObjectName() to return _objectName
- getOutputWrapper (inherited) – Get the output wrapper
- getSkippingPeek – It is an error to read from an output file
- hash (inherited) – Return a hash value for this object
- info (inherited) – Prints informational message.
- inspect (inherited) – Inspect the object.
- isEmpty – Returns TRUE_ if pipe empty
- isFilePipe (inherited) – Always returns TRUE_
- isFull – Returns TRUE_ if pipe is full
- isInputPipe (inherited) – Returns TRUE_ if this is an input pipe
- isOutputPipe – Always returns TRUE_
- notImplemented (inherited) – Print an error message that the argument list is not implemented
- peek – It is an error to peek into an output file
- printOn – Stream output function
- put – Write data to the buffer, flushing to the file when the buffer fills up.
- setByteOffset – Sets the current byte offset in the file associated with this pipe.
- setInputWrapper (inherited) – Set the input wrapper
- setOutputWrapper (inherited) – Set the output wrapper
- shouldNotImplement (inherited) – This method prints an error message that says it should not be implemented.
- subclassResponsibility (inherited) – Method prints an error message that says it should be implemented by a subclass.
- successfulOpen (inherited) – Returns true if there is a file associated with the pipe.
- truncateAfterByte – Truncates the data in the file associated with this pipe after the specified byte, leaving the seek pointer at the end of the file.
- warning (inherited) – This prints a warning message.

Included Files

"FileStreamPipe.h"

Private Methods

`void` **_flushPool**`(  );`

Flushes data in the buffer pool to the output file.

Public Methods

OutputFileStreamPipe`(  const char *fileName,`
`int bytesInWritePool = DEFAULTwritePoolSize );`

The constructor

~OutputFileStreamPipe( );

The destructor

`virtual int` **bytesFree( )** `const;`

Returns the number of bytes that can be written

`virtual int` **bytesReady( )** `const;`

Returns the number of bytes ready to read

`virtual PipeReader *` **createPipeReader( )** `const ;`

Not implemented for class

`virtual int` **get(** `char *data, int lengthRequested );`

It is an error to read from an output file

`virtual off_t` **getByteOffset( );**

Returns the current byte offset in the file associated with this pipe.

`virtual off_t` **getFileSize( );**

Returns the length of the file. Does not change the current file pointer.

`virtual int` **getSkippingPeek(** `char *data, int lengthRequested );`

It is an error to read from an output file

`virtual Boolean_` **isEmpty( )** `const;`

Returns TRUE_ if pipe empty

`virtual Boolean_` **isFull( )** `const;`

Returns TRUE_ if pipe is full

`virtual Boolean_` **isOutputPipe( )** `const ;`

Always returns TRUE_

`virtual int` **peek(** `char *data, int lengthRequested );`

It is an error to peek into an output file

`virtual int` **put(** `const char *data, int length );`

Write data to the buffer, flushing to the file when the buffer fills up.

`virtual void` **setByteOffset**`( off_t newByteOffset );`

Sets the current byte offset in the file associated with this pipe.

`virtual void` **truncateAfterByte**`( off_t byteOffset );`

Truncates the data in the file associated with this pipe after the specified byte, leaving the seek pointer at the end of the file.

`virtual void` **printOn**`( ostream& os ) const;`

Stream output function

Name

PipeReader – Definition of an Abstract Class for parsing pipe data.

Authors

Ruth A. Aydt (aydt@cs.uiuc.edu)
Contributing Author: Tom Birkett (birkett@cs.uiuc.edu)
Contributing Author: Taed Nelson (nelson@cs.uiuc.edu)

Heritage

Inherits from – Obj.

Description

A PipeReader is an abstract class defining the general methods for parsing what comes off the pipe.

Available Methods

Protected
- _setClassName (inherited) – Set _className

Public
- abort (inherited) – Print message and abort system.
- error (inherited) – This prints an error message.
- gentleWarning (inherited) – A warning message to stderr
- getAttributes – Copy pipeAttributes to attributes
- getClassName (inherited) – Return pointer to class name
- getData – Parse the record data from the interal buffer into the recordDossier. Fail if not enough data or if type is not PKT_DATA
- getDescriptor – Parse the structure descriptor from the internal buffer into structDescr. Fail if not enough data or if type is not PKT_DESCRIPTOR
- getFileSize – Returns the size of the file the pipe is connected to. Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- getObjectName (inherited) – This returns a name for the object. As defined in Obj, it returns the same string as the is returned by getClassName. However, derived classes may redefine it to include more detailed information.
- getOffset – Returns the current offset into the pipe being read from. Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- getPacketDataPtr – Return pointer to the data section of the cached packet. Beware that this pointer will become invalid after the next getPacketHeader() call
- getPacketHeader – Returns the header of the next packet in the pipe, or a header with type of PIPE_EMPTY if there is no packet. If a packet DOES exist, then it is read into the internal

buffer of the Pipe Reader. If it's type PKT_ATTRIBUTES then pipeAttributes is updated.
- hash (inherited) – Return a hash value for this object
- info (inherited) – Prints informational message.
- inspect (inherited) – Inspect the object.
- notImplemented (inherited) – Print an error message that the argument list is not implemented
- printOn – Helper function for output
- rewind – Rewind the offset to what it was when the PipeReader was created. Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- setOffset – Sets the current offset into the pipe being read from. Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- shouldNotImplement (inherited) – This method prints an error message that says it should not be implemented.
- subclassResponsibility (inherited) – Method prints an error message that says it should be implemented by a subclass.
- successfulConnection – Returns TRUE_ if the connection was successful in the constructor.
- warning (inherited) – This prints a warning message.

Included Files

<sys/types.h>
"Attributes.h"
"PacketHeader.h"
"Obj.h"
"StreamPipe.h"

Public Methods

PipeReader(`StreamPipe *thePipe` **);**

The constructor

~PipeReader(**);**

The destructor

`virtual Boolean_` **getData(** `RecordDossier& recordDossier ) = 0;`

Parse the record data from the interal buffer into the recordDossier. Fail if not enough data or if type is not PKT_DATA

`virtual Boolean_` **getDescriptor(** `StructureDescriptor& structDescr ) = 0;`

Parse the structure descriptor from the internal buffer into structDescr. Fail if not enough data or if type is not PKT_DESCRIPTOR

`virtual const char * getPacketDataPtr( ) const = 0;`

Return pointer to the data section of the cached packet. Beware that this pointer will become invalid after the next getPacketHeader() call

`virtual PacketHeader getPacketHeader( ) = 0;`

Returns the header of the next packet in the pipe, or a header with type of PIPE_EMPTY if there is no packet. If a packet DOES exist, then it is read into the internal buffer of the Pipe Reader. If it's type PKT_ATTRIBUTES then pipeAttributes is updated.

`void getAttributes( Attributes& attributes ) const;`

Copy pipeAttributes to attributes

`off_t getFileSize( );`

Returns the size of the file the pipe is connected to. Not valid for all types of pipes - if call made and invalid, program aborts with error message.

`off_t getOffset( );`

Returns the current offset into the pipe being read from. Not valid for all types of pipes - if call made and invalid, program aborts with error message.

`void rewind( );`

Rewind the offset to what it was when the PipeReader was created. Not valid for all types of pipes - if call made and invalid, program aborts with error message.

`void setOffset( off_t offset );`

Sets the current offset into the pipe being read from. Not valid for all types of pipes - if call made and invalid, program aborts with error message.

`Boolean_ successfulConnection( ) const;`

Returns TRUE_ if the connection was successful in the constructor.

`virtual void printOn( ostream& os = cout ) const;`

Helper function for output

Name

PipeWriter – Abstract Class defining general methods for outputting pipe data.

Authors

Ruth Aydt (aydt@cs.uiuc.edu)
Contributing Author: Tom Birkett (birkett@cs.uiuc.edu)
Contributing Author: Taed Nelson (nelson@cs.uiuc.edu)

Heritage

Inherits from – Obj.

Description

A PipeWriter is an abstract class defining the methods needed to output data to the pipe.

Available Methods

Protected

- _setClassName (inherited) – Set _className
- _writeSddfHeader – Write the SDDF header information to the current Stream Pipe.

Public

- abort (inherited) – Print message and abort system.
- addPipe – Add another pipe to the list we write to.
- deletePipe – Delete the pipe from the list we write to.
- error (inherited) – This prints an error message.
- gentleWarning (inherited) – A warning message to stderr
- getClassName (inherited) – Return pointer to class name
- getFileSize – Returns the size of the file the pipe is connected to. Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- getObjectName (inherited) – This returns a name for the object. As defined in Obj, it returns the same string as the is returned by getClassName. However, derived classes may redefine it to include more detailed information.
- getOffset – Returns the current offset into the pipe being written to. Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- hash (inherited) – Return a hash value for this object
- info (inherited) – Prints informational message.
- inspect (inherited) – Inspect the object.
- notImplemented (inherited) – Print an error message that the argument list is not implemented
- printOn (inherited) – Stream output function
- putAttributes – Update pipeAttributes and write it to the pipe.

- putCommand – Write the command to the pipe
- putData – Write the data associated with a record to the pipe
- putDescriptor – Write the Structure Descriptor to the pipe.
- putPacket – Write a Packet to the pipe.
- rewind – Rewind the offset to what it was when the PipeWriter was created and the header written (file is truncated to the shorter length). Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- setOffset – Sets the current offset into the pipe being read from (no change in data takes place). Not valid for all types of pipes - if call made and invalid, program aborts with error message.
- shouldNotImplement (inherited) – This method prints an error message that says it should not be implemented.
- subclassResponsibility (inherited) – Method prints an error message that says it should be implemented by a subclass.
- successfulConnection – Returns TRUE_ if the connection was successful in the constructor.
- warning (inherited) – This prints a warning message.

Included Files

<sys/types.h>
"Attributes.h"
"PacketHeader.h"
"Obj.h"

Protected Methods

`virtual void` **_writeSddfHeader**`(  ) const = 0;`

Write the SDDF header information to the current Stream Pipe.

Public Methods

PipeWriter`(  StreamPipe *thePipe );`

The constructor

~PipeWriter`(  );`

The destructor

`virtual Boolean_` **addPipe**`(  StreamPipe *thePipe );`

Add another pipe to the list we write to.

`void` **deletePipe**`(  StreamPipe *thePipe );`

Delete the pipe from the list we write to.

`virtual void` **putAttributes**`( ) = 0;`

Write pipeAttributes to the pipe.

`virtual void` **putAttributes**`( const Attributes& attributes ) = 0;`

Update pipeAttributes and write it to the pipe.

`virtual void` **putCommand**`( int pktTag = 0 ) = 0;`

Write the command to the pipe

`virtual void` **putData**`( const RecordDossier& recordDossier ) = 0;`

Write the data associated with a record to the pipe

`virtual void` **putDescriptor**`( const StructureDescriptor& structDescr,`
`int pktTag ) = 0;`

Write the Structure Descriptor to the pipe.

`virtual void` **putPacket**`( const PacketHeader& pktHeader, const char *pktData ) = 0;`

Write a Packet to the pipe.

`off_t` **getFileSize**`( );`

Returns the size of the file the pipe is connected to. Not valid for all types of pipes - if call made and invalid, program aborts with error message.

`off_t` **getOffset**`( );`

Returns the current offset into the pipe being written to. Not valid for all types of pipes - if call made and invalid, program aborts with error message.

`void` **rewind**`( );`

Rewind the offset to what it was when the PipeWriter was created and the header written (file is truncated to the shorter length). Not valid for all types of pipes - if call made and invalid, program aborts with error message.

`void` **setOffset**`( off_t offset );`

Sets the current offset into the pipe being read from (no change in data takes place). Not valid for all types of pipes - if call made and invalid, program aborts with error message.

`Boolean_` **successfulConnection**`( ) const;`

Returns TRUE_ if the connection was successful in the constructor.

Name

RecordDictionary – Class used to keep track of the record dossiers that are valid on a given input or output port.

Authors

Ruth A. Aydt (aydt.cs.uiuc.edu)

Heritage

Inherits from – Obj.

Description

There is a single instance of the RecordDictionary class for each input or output port. It is used to maintain a list of the RecordDossiers encountered by the system at that point in the processing.

Friends

Class RecordDictionaryIterator.
Class RecordDictionaryPIterator.

Available Methods

Protected
- _setClassName (inherited) – Set _className

Public
- abort (inherited) – Print message and abort system.
- contains – Membership test on match with tag
- entryCount – Returns the number of dossiers in the dictionary.
- error (inherited) – This prints an error message.
- fetch – Lookup the named tag and return the RecordDossier with that systemTag or NO-MATCH RecordDossier if no match was found.
- gentleWarning (inherited) – A warning message to stderr
- getArrayP – Get pointer to array of field with given ID in RecordDossier with specified tag. If value is not of type array, then NULL is returned.
- getClassName (inherited) – Return pointer to class name
- getObjectName (inherited) – This returns a name for the object. As defined in Obj, it returns the same string as the is returned by getClassName. However, derived classes may redefine it to include more detailed information.
- getValue – Get ref to value of field with given ID in RecordDossier with specified tag

- getValueP – Get pointer to value of field with given ID in RecordDossier with specified tag
- hash (inherited) – Return a hash value for this object
- info (inherited) – Prints informational message.
- insert – Create a new RecordDossier in the RecordDictionary. Returns Success or Failure if this is a duplicate entry
- inspect (inherited) – Inspect the object.
- notImplemented (inherited) – Print an error message that the argument list is not implemented
- printOn – Helper function for RecordDictionary output.
- setValue – Set value of field with given ID in RecordDossier with specified tag
- shouldNotImplement (inherited) – This method prints an error message that says it should not be implemented.
- subclassResponsibility (inherited) – Method prints an error message that says it should be implemented by a subclass.
- warning (inherited) – This prints a warning message.

Included Files

"Defines.h"
"Obj.h"
"RecordDossier.h"
"StructureDescriptor.h"

Public Methods

RecordDictionary( );

The RecordDictionary constructor with no initialization

~RecordDictionary( );

The RecordDictionary destructor

`Boolean_` **contains(** `int dossierTag ) const;`

Membership test on match with tag

`int` **entryCount(** `) const;`

Returns the number of dossiers in the dictionary.

`RecordDossier&` **fetch(** `int dossierTag ) const;`

Lookup the named tag and return the RecordDossier with that systemTag or NOMATCH RecordDossier if no match was found.

```
Array * getArrayP(   int dossierTag,  const CString& fieldName ) const;
```

Get pointer to array of named field in RecordDossier with specified tag. If value is not of type array, then NULL pointer returned.

```
Array * getArrayP(   int dossierTag,  int fieldID ) const;
```

Get pointer to array of field with given ID in RecordDossier with specified tag. If value is not of type array, then NULL is returned.

```
Value& getValue(   int dossierTag,  const CString& fieldName ) const;
```

Get ref to value of named field in RecordDossier with specified tag

```
Value& getValue(   int dossierTag,  int fieldID ) const;
```

Get ref to value of field with given ID in RecordDossier with specified tag

```
Value * getValueP(   int dossierTag,  const CString& fieldName ) const;
```

Get pointer to value of named field in RecordDossier with specified tag

```
Value * getValueP(   int dossierTag,  int fieldID ) const;
```

Get pointer to value of field with given ID in RecordDossier with specified tag

```
Boolean_ insert(   int dossierTag,  const StructureDescriptor& structDescr );
```

Create a new RecordDossier in the RecordDictionary. Returns Success or Failure if this is a duplicate entry

```
Boolean_ setValue(   int dossierTag,  const CString& fieldName,
                     const Value& fieldValue);
```

Set value of named field in RecordDossier with specified tag.

```
Boolean_ setValue(   int dossierTag,  int fieldID,  const Value& fieldValue );
```

Set value of field with given ID in RecordDossier with specified tag

```
virtual void printOn(   ostream& strm = cout ) const;
```

Helper function for RecordDictionary output.

Name

RecordDictionaryPIterator – Class used to step through the RecordDossier entries in a RecordDictionary returning pointers to the Dossiers. See also RecordDictionaryIterator which returns references to the Dossiers.

Authors

Ruth A. Aydt (aydt.cs.uiuc.edu)

Heritage

Inherits from – Obj.

Description

A RecordDictionaryPIterator is used to step through the entries in a RecordDictionary. It's member variables maintain a sense of the "current" RecordDictEntry in the Dictionary.

Available Methods

Protected

- _setClassName (inherited) – Set _className

Public

- abort (inherited) – Print message and abort system.
- error (inherited) – This prints an error message.
- first – Returns pointer to first RecordDossier in the RecordDictionary or NULL if there isn't one.
- gentleWarning (inherited) – A warning message to stderr
- getClassName (inherited) – Return pointer to class name
- getObjectName (inherited) – This returns a name for the object. As defined in Obj, it returns the same string as the is returned by getClassName. However, derived classes may redefine it to include more detailed information.
- hash (inherited) – Return a hash value for this object
- info (inherited) – Prints informational message.
- inspect (inherited) – Inspect the object.
- next – Returns pointer to next RecordDossier in the RecordDictionary or NULL if there isn't one. If called before first() for a particular instance of the iterator it behaves the same as first().
- notImplemented (inherited) – Print an error message that the argument list is not implemented
- printOn (inherited) – Stream output function
- shouldNotImplement (inherited) – This method prints an error message that says it should not be implemented.

- subclassResponsibility (inherited) – Method prints an error message that says it should be implemented by a subclass.
- warning (inherited) – This prints a warning message.

Included Files

"Obj.h"
"RecordDictionary.h"
"RecordDossier.h"

Public Methods

RecordDictionaryPIterator(`const RecordDictionary& recordDict` **);**

The PIterator constructor with reference to RecordDictionary.

~RecordDictionaryPIterator( );

The PIterator destructor

`RecordDossier *` **first( );**

Returns pointer to first RecordDossier in the RecordDictionary or NULL if there isn't one.

`RecordDossier *` **next( );**

Returns pointer to next RecordDossier in the RecordDictionary or NULL if there isn't one. If called before first() for a particular instance of the iterator it behaves the same as first().

Name

RecordDossier – Class used to cache structure information gleaned from the StructureDescriptor and provide space and methods for associating data packets with their StructureDescriptors.

Authors

Ruth A. Aydt (aydt@cs.uiuc.edu)
Contributing Author: Tom Birkett (birkett@cs.uiuc.edu)

Heritage

Inherits from – BaseDescriptor, Obj.

Description

A RecordDossier is used within the system to store and manipulate the actual data. It caches the structure descriptor information and record data. The record data fields will initially be empty. The RecordDossier actually contains redundant information that could be derived from elsewhere in the system. It exists to simplify and speed up processing.

Available Methods

Protected

- _setClassName (inherited) – Set _className

Public

- abort (inherited) – Print message and abort system.
- bitsToObj – Set the field values using data in the bitstream pointed to by dataPtr. Return the number of bytes that were needed to set values or -1 if an error occured.
- bytesNeeded – Bytes necessary to represent the field values in binary bitstream format
- cbitsToObj – Set the field values using data in the non-native bitstream pointed to by dataPtr. Return the number of bytes that were needed to set values or -1 if an error occured.
- cbytesNeeded – Bytes necessary to represent the field values in non-native binary bitstream format
- entryCount – Return the number of fields in the Dossier
- error (inherited) – This prints an error message.
- gentleWarning (inherited) – A warning message to stderr
- getArrayP – Get pointer to Array of named field. Return NULL if value is not an array.
- getAttributes (inherited) – Returns the attribute dictionary for this component
- getClassName (inherited) – Return pointer to class name
- getField – Get pointer to FieldDescriptor with given ID
- getFieldID – Return the ID of the field with the given name. Returns -1 if no match.
- getName (inherited) – Returns the name of the component

- getObjectName (inherited) – Overloads Obj::getObjectName() to return the full name including class and component names.
- getTag – Get the tag
- getValue – Get value of field with given ID
- getValueP – Get pointer to Value of field with given ID
- hash (inherited) – Return a hash value for this object
- info (inherited) – Prints informational message.
- insertAttributes (inherited) – Insert a new attribute
- inspect (inherited) – Inspect the object.
- notImplemented (inherited) – Print an error message that the argument list is not implemented
- notNull (inherited) – Returns FALSE_ if name is NOMATCH, otherwise returns TRUE_.
- objToBits – Use the field values to populate the binary bitstream pointed to by bufPtr. Return the number of bytes actually used or -1 if bufLen was too small to hold values.
- objToCbits – Use the field values to populate the non-native binary bitstream pointed to by bufPtr. Return the number of bytes actually used or -1 if bufLen was too small to hold values.
- operator= – Assigment with copying of data
- printOn – Helper function for RecordDossier output.
- setAttributes (inherited) – Set the attribute dictionary for this component
- setName (inherited) – Set the name of the component. Also update objectName;
- setValue – Set value of field with given ID. Returns FAILURE_ if fieldID invalid or fieldValue.traits cannot sensibly be converted to match descriptor->traits
- shouldNotImplement (inherited) – This method prints an error message that says it should not be implemented.
- subclassResponsibility (inherited) – Method prints an error message that says it should be implemented by a subclass.
- warning (inherited) – This prints a warning message.

Included Files

"BaseDescriptor.h"
"Defines.h"
"FieldDescriptor.h"
"Value.h"

Public Methods

RecordDossier(`const CString& dossierName );`

This constructor takes one argument - the dossier name. It is used to build a "dummy" RecordDossier with tag = 0 and fieldCount = 0.

RecordDossier(`const RecordDossier& originalDossier );`

This constructor provides copy semantics for RecordDossier objects. That is, it creates a RecordDossier which contains the same information as its argument, but shares no pointers.

RecordDossier(`int sysTag, const StructureDescriptor& structureDescr );`

The constructor specifying a tag and StructureDescriptor. NOTE: This is usually called by Record-Dictionary:: insert()

~RecordDossier(`);`

The RecordDossier destructor

`int` **bitsToObj(** `const char *dataPtr );`

Set the field values using data in the bitstream pointed to by dataPtr. Return the number of bytes that were needed to set values or -1 if an error occured.

`int` **bytesNeeded(** `) const;`

Bytes necessary to represent the field values in binary bitstream format

`int` **cbitsToObj(** `const char *dataPtr, const DataCharacteristics *cnvInfo );`

Set the field values using data in the non-native bitstream pointed to by dataPtr. Return the number of bytes that were needed to set values or -1 if an error occured.

`int` **cbytesNeeded(** `const DataCharacteristics *cnvInfo ) const;`

Bytes necessary to represent the field values in non-native binary bitstream format

`int` **entryCount(** `) const;`

Return the number of fields in the Dossier

`Array *` **getArrayP(** `const CString& fieldName ) const;`

Get pointer to Array of named field. Return NULL if value is not an array.

`Array *` **getArrayP(** `int fieldID ) const;`

Get pointer to array of field with given ID. Return NULL if value is not an array

`FieldDescriptor *` **getField(** `const CString& fieldName ) const;`

Get pointer to named FieldDescriptor

`FieldDescriptor *` **getField(** `int fieldID ) const;`

Get pointer to FieldDescriptor with given ID

`int` **getFieldID(** `const CString& fieldName ) const;`

Return the ID of the field with the given name. Returns -1 if no match.

`int` **getTag**`( ) const;`

Get the tag

`Value&` **getValue**`( const CString& fieldName ) const;`

Get value of named field

`Value&` **getValue**`( int fieldID ) const;`

Get value of field with given ID

`Value *` **getValueP**`( const CString& fieldName ) const;`

Get pointer to Value of named field

`Value *` **getValueP**`( int fieldID ) const;`

Get pointer to Value of field with given ID

`int` **objToBits**`( char *const bufPtr, int bufLen ) const;`

Use the field values to populate the binary bitstream pointed to by bufPtr. Return the number of bytes actually used or -1 if bufLen was too small to hold values.

`int` **objToCbits**`( char *const bufPtr, int bufLen,`
`const DataCharacteristics *cnvInfo ) const;`

Use the field values to populate the non-native binary bitstream pointed to by bufPtr. Return the number of bytes actually used or -1 if bufLen was too small to hold values.

`Boolean_` **setValue**`( const CString& fieldName, const Value& fieldValue );`

Set value of named field. Returns FAILURE_ if field not found or if fieldValue.traits cannot sensibly be converted to match descriptor->traits

`Boolean_` **setValue**`( int fieldID, const Value& fieldValue );`

Set value of field with given ID. Returns FAILURE_ if fieldID invalid or fieldValue.traits cannot sensibly be converted to match descriptor->traits

`RecordDossier&` **operator**`=( const RecordDossier& recordDossier );`

Assigment with copying of data

`virtual void` **printOn**`( ostream& strm = cout ) const;`

Helper function for RecordDossier output.

Name

StreamPipe – Abstract base class for stream-based pipes.

Authors

Robert Olson (olson@cs.uiuc.edu)
Contributing Author: Ruth A. Aydt (aydt@cs.uiuc.edu)

Heritage

Inherits from – BasePipe, Obj.

Description

A StreamPipe supports stream-based communication (ie the passing of blocks of data without explicit record boundaries).

Available Methods

Protected

- _setClassName (inherited) – Set _className
- _setObjectName (inherited) – Give detailed name to Pipe

Public

- abort (inherited) – Print message and abort system.
- bytesFree – Return the number of bytes that can be written
- bytesReady – Return the number of bytes ready to be read
- createPipeReader – Implemented by derived classes to create appropriate type of PipeReader for the StreamPipe. Returns NULL if can't create PipeReader.
- error (inherited) – This prints an error message.
- gentleWarning (inherited) – A warning message to stderr
- get – Get data from the pipe
- getByteOffset – Returns the current byte offset in the pipe. This is not valid for all types of pipes. Implemented in appropriate derived classes.
- getClassName (inherited) – Return pointer to class name
- getFileSize – Returns the length of the file. Does not change the current file pointer. This is not valid for all types of pipes. Implemented in appropriate derived classes.
- getInputCharacteristics – Return pointer to the data characteristics for the input associated with this pipe. Only valid if inputType is CONVERT_BINARY_SDDF. Implemented in appropriate derived classes.
- getInputType – Return the type of the input associated with this pipe. Returns valid type in appropriate derived classes.
- getInputWrapper (inherited) – Get the input wrapper

- getObjectName (inherited) – Overloads Obj::getObjectName() to return _objectName
- getOutputWrapper (inherited) – Get the output wrapper
- getSkippingPeek – Get data from the pipe, skipping over data in the peek buffer. Also clear peek buffer.
- hash (inherited) – Return a hash value for this object
- info (inherited) – Prints informational message.
- inspect (inherited) – Inspect the object.
- isEmpty – Returns TRUE_ if the pipe is empty
- isFilePipe (inherited) – Returns TRUE_ if this is a file pipe
- isFull – Returns TRUE_ if the pipe is full
- isInputPipe (inherited) – Returns TRUE_ if this is an input pipe
- isOutputPipe (inherited) – Returns TRUE_ if this is an output pipe
- notImplemented (inherited) – Print an error message that the argument list is not implemented
- peek – Peek at data in the pipe
- printOn – Stream output function
- put – Write data into the pipe
- setByteOffset – Sets the current byte offset in the pipe. This is not valid for all types of pipes. Implemented in appropriate derived classes.
- setInputWrapper (inherited) – Set the input wrapper
- setOutputWrapper (inherited) – Set the output wrapper
- shouldNotImplement (inherited) – This method prints an error message that says it should not be implemented.
- subclassResponsibility (inherited) – Method prints an error message that says it should be implemented by a subclass.
- truncateAfterByte – Truncates the data in the pipe after the specified byte. This is only valid for pipes associated with output data files.
- warning (inherited) – This prints a warning message.

Included Files

<sys/types.h>
"BasePipe.h"

Public Methods

StreamPipe( );

The StreamPipe constructor

~StreamPipe( );

The StreamPipe destructor

`virtual int` **bytesFree**`( ) const = 0;`

Return the number of bytes that can be written

`virtual int` **bytesReady**`( ) const = 0;`

Return the number of bytes ready to be read

`virtual PipeReader *` **createPipeReader**`( ) const = 0;`

Implemented by derived classes to create appropriate type of PipeReader for the StreamPipe. Returns NULL if can't create PipeReader.

`virtual int` **get**`( char *data, int lengthRequested ) = 0;`

Get data from the pipe

`virtual off_t` **getByteOffset**`( ) ;`

Returns the current byte offset in the pipe. This is not valid for all types of pipes. Implemented in appropriate derived classes.

`virtual off_t` **getFileSize**`( ) ;`

Returns the length of the file. Does not change the current file pointer. This is not valid for all types of pipes. Implemented in appropriate derived classes.

`virtual const DataCharacteristics *` **getInputCharacteristics**`( ) const ;`

Return pointer to the data characteristics for the input associated with this pipe. Only valid if inputType is CONVERT_BINARY_SDDF. Implemented in appropriate derived classes.

`virtual PabloSddfType` **getInputType**`( ) const ;`

Return the type of the input associated with this pipe. Returns valid type in appropriate derived classes.

`virtual int` **getSkippingPeek**`( char *data, int lengthRequested ) = 0;`

Get data from the pipe, skipping over data in the peek buffer. Also clear peek buffer.

`virtual Boolean_` **isEmpty**`( ) const = 0;`

Returns TRUE_ if the pipe is empty

`virtual int` **peek**`( char *data, int lengthRequested ) = 0;`

Peek at data in the pipe

`virtual int` **put**`( const char *data, int length ) = 0;`

Write data into the pipe

`virtual Boolean_` **isFull**`( ) const = 0;`

Returns TRUE_ if the pipe is full

`virtual void` **setByteOffset**`( off_t ) ;`

Sets the current byte offset in the pipe. This is not valid for all types of pipes. Implemented in appropriate derived classes.

`virtual void` **truncateAfterByte**`( off_t ) ;`

Truncates the data in the pipe after the specified byte. This is only valid for pipes associated with output data files.

`virtual void` **printOn**`( ostream& os ) const;`

Stream output function

Name

StructureDescriptor – Class used to describe a structure made up of one or more fields of data in the self-documenting data format.

Authors

Ruth A. Aydt (aydt.cs.uiuc.edu)

Heritage

Inherits from – BaseDescriptor, Obj.

Description

A StructureDescriptor defines the smallest unit of data which can be passed through the system. Each StructureDescriptor groups one or more FieldDescriptors into a logical element of the self-documenting data format. Within a StructureDescriptor instance, each field.name must be unique.

Friends

Class StructureDescriptorIterator.

Available Methods

Protected

- _setClassName (inherited) – Set _className

Public

- abort (inherited) – Print message and abort system.
- bitsToObj – Translates from binary bitstream representation of data to StructureDescriptor instance. Any fields in the instance when the call is made are deleted. Returns number of bytes read or -1 if error occured.
- bytesNeeded – Returns the number of bytes needed to represent instance as a binary bitstream.
- cbitsToObj – Translates from non-native binary bitstream representation of data to StructureDescriptor instance. Any fields in the instance when the call is made are deleted. Returns number of bytes read or -1 if error occured.
- cbytesNeeded – Returns the number of bytes needed to represent instance as a non-native binary bitstream.
- contains – Field membership test
- entryCount – Returns the number of field entries in this StructureDescriptor
- error (inherited) – This prints an error message.

- fetch – Lookup the named field and return the FieldDescriptor of that name, or a NOMATCH FieldDescriptor if no match was found.
- gentleWarning (inherited) – A warning message to stderr
- getAttributes (inherited) – Returns the attribute dictionary for this component
- getClassName (inherited) – Return pointer to class name
- getName (inherited) – Returns the name of the component
- getObjectName (inherited) – Overloads Obj::getObjectName() to return the full name including class and component names.
- hash (inherited) – Return a hash value for this object
- info (inherited) – Prints informational message.
- insert – Insert a FieldDescriptor into the StructureDescriptor. Returns SUCCESS or FAILURE (if duplicate field.name)
- insertAttributes (inherited) – Insert a new attribute
- inspect (inherited) – Inspect the object.
- isMatchFor – Checks to see if structureDescr matches this StructureDescriptor at the level specified by matchOn {Name, Attributes, Fields}
- notImplemented (inherited) – Print an error message that the argument list is not implemented
- notNull (inherited) – Returns FALSE_ if name is NOMATCH, otherwise returns TRUE_.
- objToBits – Translates StructureDescriptor instance to binary bitstream representation of data. Returns number of bytes used to write bitstream, or -1 if bufLen $<$ bytes needed to represent data.
- objToCbits – Translates StructureDescriptor instance to non-native binary bitstream representation of data. Returns number of bytes used to write bitstream, or -1 if bufLen $<$ bytes needed to represent data.
- operator= – StructureDescriptor assigment
- printOn – Helper function for StructureDescriptor output.
- remove – Remove the named field from the StructureDescriptor.
- setAttributes (inherited) – Set the attribute dictionary for this component
- setName (inherited) – Set the name of the component. Also update objectName;
- shouldNotImplement (inherited) – This method prints an error message that says it should not be implemented.
- subclassResponsibility (inherited) – Method prints an error message that says it should be implemented by a subclass.
- warning (inherited) – This prints a warning message.

Included Files

"BaseDescriptor.h"
"Defines.h"
"FieldDescriptor.h"

Public Methods

StructureDescriptor(**);**

A StructureDescriptor constructor with no initialization

StructureDescriptor(`const CString& structureName` **);**

A StructureDescriptor constructor with initial value for name

StructureDescriptor(`const CString& structureName,`
`const Attributes& structureAttributes` **);**

A StructureDescriptor constructor with initial values for name and attributes

StructureDescriptor(`const StructureDescriptor& structureDescr` **);**

StructureDescriptor constructor with copy initialization

~StructureDescriptor(**);**

The StructureDescriptor destructor

`int` **bitsToObj(** `const char *bufPtr` **);**

Translates from binary bitstream representation of data to StructureDescriptor instance. Any fields in the instance when the call is made are deleted. Returns number of bytes read or -1 if error occured.

`int` **bytesNeeded(** **)** `const;`

Returns the number of bytes needed to represent instance as a binary bitstream.

`int` **cbitsToObj(** `const char *bufPtr, const DataCharacteristics *cnvInfo` **);**

Translates from non-native binary bitstream representation of data to StructureDescriptor instance. Any fields in the instance when the call is made are deleted. Returns number of bytes read or -1 if error occured.

`int` **cbytesNeeded(** `const DataCharacteristics *cnvInfo` **)** `const;`

Returns the number of bytes needed to represent instance as a non-native binary bitstream.

`Boolean_` **contains(** `const CString& fieldName` **)** `const;`

Field membership test

`int` **entryCount(** **)** `const;`

Returns the number of field entries in this StructureDescriptor

```
const FieldDescriptor& fetch(   const CString& fieldName ) const;
```

Lookup the named field and return the FieldDescriptor of that name, or a NOMATCH FieldDescriptor if no match was found.

```
Boolean_ insert(   const FieldDescriptor& fieldDescr );
```

Insert a FieldDescriptor into the StructureDescriptor. Returns SUCCESS or FAILURE (if duplicate field.name)

```
Boolean_ isMatchFor(   const StructureDescriptor& structureDescr,
                       Boolean_ matchOnName,  Boolean_ matchOnAttibutes,
                       Boolean_ matchOnFields ) const;
```

Checks to see if structureDescr matches this StructureDescriptor at the level specified by matchOn {Name, Attributes, Fields}

```
int objToBits(   char* const bufPtr,  int bufLen ) const;
```

Translates StructureDescriptor instance to binary bitstream representation of data. Returns number of bytes used to write bitstream, or -1 if bufLen $<$ bytes needed to represent data.

```
int objToCbits(   char* const bufPtr,  int bufLen,
                  const DataCharacteristics *cnvInfo ) const;
```

Translates StructureDescriptor instance to non-native binary bitstream representation of data. Returns number of bytes used to write bitstream, or -1 if bufLen $<$ bytes needed to represent data.

```
void remove(   const CString& fieldName );
```

Remove the named field from the StructureDescriptor.

```
StructureDescriptor& operator=(   const StructureDescriptor& structureDescr);
```

StructureDescriptor assigment

```
virtual void printOn(   ostream& strm = cout ) const;
```

Helper function for StructureDescriptor output.

Name

StructureDescriptorIterator – Class used to step through fields in a StructureDescriptor.

Authors

Ruth A. Aydt (aydt.cs.uiuc.edu)

Heritage

Inherits from – Obj.

Description

A StructureDescriptorIterator is used to step through the fields in a StructureDesciptor. It's member variables maintain a sense of the "current" StructureEntry in the Descriptor.

Available Methods

Protected

- _setClassName (inherited) – Set _className

Public

- abort (inherited) – Print message and abort system.
- error (inherited) – This prints an error message.
- first – Returns reference to first FieldDescriptor in the StructureDescriptor or to "noField" if there isn't one.
- gentleWarning (inherited) – A warning message to stderr
- getClassName (inherited) – Return pointer to class name
- getObjectName (inherited) – This returns a name for the object. As defined in Obj, it returns the same string as the is returned by getClassName. However, derived classes may redefine it to include more detailed information.
- hash (inherited) – Return a hash value for this object
- info (inherited) – Prints informational message.
- inspect (inherited) – Inspect the object.
- next – Returns reference to next FieldDescriptor in the StructureDescriptpor or to "noField" if there isn't one. If called before first() for a particular instance of the iterator it behaves the same as first().
- notImplemented (inherited) – Print an error message that the argument list is not implemented
- printOn (inherited) – Stream output function
- shouldNotImplement (inherited) – This method prints an error message that says it should not be implemented.
- subclassResponsibility (inherited) – Method prints an error message that says it should be

implemented by a subclass.
- warning (inherited) – This prints a warning message.

Included Files

"FieldDescriptor.h"
"Obj.h"
"StructureDescriptor.h"

Public Methods

StructureDescriptorIterator(`const StructureDescriptor& structDescr` **);**

The Iterator constructor with reference to StructureDescriptor.

~StructureDescriptorIterator( );

The Iterator destructor

`const FieldDescriptor&` **first( );**

Returns reference to first FieldDescriptor in the StructureDescriptor or to "noField" if there isn't one.

`const FieldDescriptor&` **next( );**

Returns reference to next FieldDescriptor in the StructureDescriptpor or to "noField" if there isn't one. If called before first() for a particular instance of the iterator it behaves the same as first().

Name

Value – Header file for class used to define common methods for the various data types in the system.

Authors

Ruth A. Aydt (aydt@cs.uiuc.edu)
Contributing Author: Brian K. Totty (totty@cs.uiuc.edu)

Description

The Value Class provides a uniform interface to data of different types.

Available Methods

Public

- Array* – Conversion to Array*
- bitsToObj – Set the value from binary bitstream representation of data. The bitstream data should match the traits currently associated with this value. Return number of bytes used or -1 if error
- bytesNeeded – Returns number of Bytes needed to represent this value as a binary bitstream
- cbitsToObj – Set the value from binary bitstream representation of data - converting to native representation. The bitstream data should match the traits currently associated with this value. Return number of bytes used or -1 if error
- cbytesNeeded – Returns number of Bytes needed to represent this value as a non-native binary bitstream
- char – Conversion to char
- double – Conversion to double
- float – Conversion to float
- getResultType – Returns the data type of the Value resulting from an arithmetic operation on this Value and another with the MachineDataType specified.
- getTraits – Returns the data traits of the Value
- getValueAs – Returns Value with specified type
- getValueAs – Sets newValue to have specified type and same data as this value
- int – Conversion to int
- isDefined – Returns TRUE_ if Value is defined.
- isUndefined – Returns TRUE_ if Value is undefined.
- objToBits – Represent the value as a binary bitstream. Return number of bytes used, or -1 if there was not enough space available to hold the data.
- objToCbits – Represent the value as a non-native binary bitstream. Return number of bytes used, or -1 if there was not enough space to hold the data.
- operator!= – Value "not equal to" comparison. Undefined when type == UNDEFINED or dimension != 0

- operator+ – Value addition - undefined when type = UNDEFINED or dimension != 0
- operator< – Value "less than" comparison. Undefined when type == UNDEFINED or dimension != 0
- operator<= – Value "less than or equal to" comparison. Undefined when type == UNDEFINED or dimension != 0
- operator= – Array pointer to Value assignment. Upon completion, this Value instance points to it's own copy of the Array data.
- operator= – Value to Value assignment (upon completion both Value instances have copies of the data)
- operator= – char to Value assignment
- operator= – double to Value assignment
- operator= – float to Value assignment
- operator= – integer to Value assignment
- operator== – Value "equal to" comparison. Undefined when type == UNDEFINED or dimension != 0
- operator> – Value "greater than" comparison. Undefined when type == UNDEFINED or dimension != 0
- operator>= – Value "greater than or equal to" comparison. Undefined when type == UNDEFINED or dimension != 0
- operator* – Value multiplication - undefined when type = UNDEFINED or dimension != 0
- operator- – Value "unary minus" operator. Undefined when type == UNDEFINED or dimension != 0
- operator- – Value subtraction - undefined when type = UNDEFINED or dimension != 0
- operator/ – Value division - undefined when type = UNDEFINED or dimension != 0
- printOn – Helper function for Value output
- setQuantity – Method which sets the quantity of this Value to the quantity of newValue without changing type of this Value. If the type of the newValue is such that it cannot be cast into the type of this Value without losing precision, Value is not changed and FAILURE_ is returned. This method will work for Values of any dimension as long as the dimension of newValue is the same as the dimension of this Value.(See also the assignment operators to change both quantity and type of this Value.)

Included Files

"Array.h"
"DataTraits.h"
"Obj.h"

Public Methods

Value( );

Constructor with no arguments. Will default to UNDEFINED type and dimension 0.

Value(`MachineDataType _type, int _dimension );`

Constructor with type and dimension specified

Value(`const DataTraits& _traits );`

Constructor with DataTraits specified

Value(`const Value& value );`

Constructor with copy initialization

Value(`char dataValue );`

Constructor with character data

Value(`int dataValue );`

Constructor with integer data

Value(`float dataValue );`

Constructor with float data

Value(`double dataValue );`

Constructor with double data

Value(`Array *dataValue );`

Constructor with pointer to Array instance. The ctor will actually create a new Array instance and the pointer to that is kept in the Value. This insures that Value can free the Array it references when it is destroyed without having to worry about leaving dangling references.

~Value( );

The Value Destructor

`int` **bitsToObj(** `const char *bufPtr );`

Set the value from binary bitstream representation of data. The bitstream data should match the traits currently associated with this value. Return number of bytes used or -1 if error

`int` **bytesNeeded(** `) const;`

Returns number of Bytes needed to represent this value as a binary bitstream

`int` **cbitsToObj(** `const char *bufPtr, const DataCharacteristics *cnvInfo );`

Set the value from binary bitstream representation of data - converting to native representation. The bitstream data should match the traits currently associated with this value. Return number of bytes used or -1 if error

`int` **cbytesNeeded**`( const DataCharacteristics *cnvInfo ) const;`

Returns number of Bytes needed to represent this value as a non-native binary bitstream

`MachineDataType` **getResultType**`( MachineDataType operand_type ) const;`

Returns the data type of the Value resulting from an arithmetic operation on this Value and another with the MachineDataType specified.

`const DataTraits&` **getTraits**`( ) const;`

Returns the data traits of the Value

`Value` **getValueAs**`( MachineDataType type ) const;`

Returns Value with specified type

`void` **getValueAs**`( MachineDataType type, Value& newValue ) const;`

Sets newValue to have specified type and same data as this value

`Boolean_` **isDefined**`( ) const;`

Returns TRUE_ if Value is defined.

`Boolean_` **isUndefined**`( ) const;`

Returns TRUE_ if Value is undefined.

`int` **objToBits**`( char* const bufPtr, int bufSize ) const;`

Represent the value as a binary bitstream. Return number of bytes used, or -1 if there was not enough space available to hold the data.

```
int objToCbits( char* const bufPtr, int bufSize,
                const DataCharacteristics *cnvInfo ) const;
```

Represent the value as a non-native binary bitstream. Return number of bytes used, or -1 if there was not enough space to hold the data.

`Boolean_` **setQuantity**`( const Value& newValue );`

Method which sets the quantity of this Value to the quantity of newValue without changing type of this Value. If the type of the newValue is such that it cannot be cast into the type of this Value without losing precision, Value is not changed and FAILURE_ is returned. This method will work for Values of any dimension as long as the dimension of newValue is the same as the dimension of this Value.(See also the assignment operators to change both quantity and type of this Value.)

`Value&` **operator=**`( const Value& value );`

Value to Value assignment (upon completion both Value instances have copies of the data)

Value& operator=(**char dataValue);**

char to Value assignment

Value& operator=(**int dataValue);**

integer to Value assignment

Value& operator=(**float dataValue);**

float to Value assignment

Value& operator=(**double dataValue);**

double to Value assignment

Value& operator=(**const Array *dataValue);**

Array pointer to Value assignment. Upon completion, this Value instance points to it's own copy of the Array data.

Boolean_ operator<(**const Value& v);**

Value "less than" comparison. Undefined when type == UNDEFINED or dimension != 0

Boolean_ operator>(**const Value& v);**

Value "greater than" comparison. Undefined when type == UNDEFINED or dimension != 0

Boolean_ operator<=(**const Value& v);**

Value "less than or equal to" comparison. Undefined when type == UNDEFINED or dimension != 0

Boolean_ operator>=(**const Value& v);**

Value "greater than or equal to" comparison. Undefined when type == UNDEFINED or dimension != 0

Boolean_ operator==(**const Value& v);**

Value "equal to" comparison. Undefined when type == UNDEFINED or dimension != 0

Boolean_ operator!=(**const Value& v);**

Value "not equal to" comparison. Undefined when type == UNDEFINED or dimension != 0

Value operator-(**);**

Value "unary minus" operator. Undefined when type == UNDEFINED or dimension != 0

Value operator+(**const Value& v);**

Value addition - undefined when type = UNDEFINED or dimension != 0

```
Value operator-(   const Value& v );
```
Value subtraction - undefined when type = UNDEFINED or dimension != 0

```
Value operator*(   const Value& v );
```
Value multiplication - undefined when type = UNDEFINED or dimension != 0

```
Value operator/(   const Value& v );
```
Value division - undefined when type = UNDEFINED or dimension != 0

```
operator char(   ) const;
```
Conversion to char

```
operator int(   ) const;
```
Conversion to int

```
operator float(   ) const;
```
Conversion to float

```
operator double(   ) const;
```
Conversion to double

```
operator Array*(   ) const;
```
Conversion to Array*

```
void printOn(   ostream& strm = cout ) const;
```
Helper function for Value output

Pablo Class Hierarchy

Array 1
CString 32
DataTraits 47
Obj 61
    Attributes 15
    AttributesIterator 19
    BaseDescriptor 21
        FieldDescriptor 50
        RecordDossier 79
        StructureDescriptor 87
    BasePipe 23
        StreamPipe 83
            FileStreamPipe 54
                InputFileStreamPipe 57
                OutputFileStreamPipe 64
    DataCharacteristics 41
    PipeReader 68
        AsciiPipeReader 7
        BinaryPipeReader 26
        ConversionPipeReader 35
    PipeWriter 71
        AsciiPipeWriter 11
        BinaryPipeWriter 29
        ConversionPipeWriter 38
    RecordDictionary 74
    RecordDictionaryPIterator 77
    StructureDescriptorIterator 91
Value 93

Index to Classes and their Methods

Array, 1
- Array, 3
- ~Array, 3
- atomSize, 3
- bitsToObj, 3
- bytesNeeded, 3
- cbitsToObj, 3
- cbytesNeeded, 3
- clearDelta, 3
- getCellCount, 3
- getCellValue, 3, 4
- getDeltaCount, 4
- getDeltaValue, 4
- getDimension, 4
- getDimSizes, 4
- getTheCellValue, 4
- getType, 4
- isDeltaArray, 5
- objToBits, 5
- objToCbits, 5
- operator=, 6
- printOn, 6
- resetDelta, 5
- resetIterateCount, 5
- setCellString, 5
- setCellValue, 5
- setCellValues, 5
- setDeltaIndex, 6
- setDimSizes, 5
- setTheCellValue, 5

AsciiPipeReader, 7
- _badInput, 8
- _getArray, 8
- _getAttributes, 9
- _getCommandTag, 9
- _getDataTag, 9
- _getDescriptorTag, 9
- _getField, 9
- _getPipeAttributes, 9
- _getString, 9
- _getType, 9
- _getValue, 9
- _readPacket, 9
- _skipWhiteSpace, 9
- AsciiPipeReader, 10
- ~AsciiPipeReader, 10
- getData, 10
- getDescriptor, 10
- getPacketDataPtr, 10
- getPacketHeader, 10

AsciiPipeWriter, 11
- _checkBufferSize, 12
- _increaseBufferSize, 12
- _putArray, 12
- _putC, 13
- _putValue, 13
- _writeCString, 13
- _writePacket, 13
- _writeSddfHeader, 13
- addPipe, 13
- AsciiPipeWriter, 13
- ~AsciiPipeWriter, 13
- putAttributes, 13
- putCommand, 13
- putData, 13
- putDescriptor, 14
- putPacket, 14

Attributes, 15
- Attributes, 16
- ~Attributes, 17
- bitsToObj, 17
- bytesNeeded, 17
- cbitsToObj, 17
- cbytesNeeded, 17
- clearEntries, 17
- contains, 17
- entryCount, 17
- fetch, 17
- insert, 17

objToBits, 17
objToCbits, 18
operator!=, 18
operator=, 18
operator==, 18
operator[], 18
printOn, 18
remove, 18

AttributesIterator, 19
AttributesIterator, 20
~AttributesIterator, 20
first, 20
next, 20
theValue, 20

BaseDescriptor, 21
BaseDescriptor, 22
~BaseDescriptor, 22
getAttributes, 22
getName, 22
getObjectName, 22
insertAttributes, 22
notNull, 22
printOn, 22
setAttributes, 22
setName, 22

BasePipe, 23
_setObjectName, 24
BasePipe, 24
~BasePipe, 24
getInputWrapper, 24
getObjectName, 24
getOutputWrapper, 24
isFilePipe, 24
isInputPipe, 24
isOutputPipe, 24
printOn, 25
setInputWrapper, 24
setOutputWrapper, 25

BinaryPipeReader, 26
_readPacket, 27
BinaryPipeReader, 27
~BinaryPipeReader, 27
getData, 27
getDescriptor, 27
getPacketDataPtr, 27
getPacketHeader, 28

BinaryPipeWriter, 29
_writePacket, 30
_writeSddfHeader, 30
BinaryPipeWriter, 30
~BinaryPipeWriter, 30
putAttributes, 30, 31
putCommand, 31
putData, 31
putDescriptor, 31
putPacket, 31

ConversionPipeReader, 35
_readPacket, 36
ConversionPipeReader, 36
~ConversionPipeReader, 36
getData, 36
getDescriptor, 36
getPacketDataPtr, 37
getPacketHeader, 37

ConversionPipeWriter, 38
_writePacket, 39
_writeSddfHeader, 39
addPipe, 40
ConversionPipeWriter, 39
~ConversionPipeWriter, 40
putAttributes, 40
putCommand, 40
putData, 40
putDescriptor, 40
putPacket, 40

CString, 32
char*, 34
concat, 33
CString, 33
~CString, 33

getValue, 33
length, 33
operator!=, 34
operator+, 34
operator<=, 34
operator=, 33
operator==, 34
operator>=, 34
operator[], 34
printOn, 34

DataCharacteristics, 41
DataCharacteristics, 43
functionGet, 43
functionPut, 43, 44
getAtomSize, 44
initializeFromFile, 44
intSize, 44
notNative, 44
setByteOrdering, 44
setByteOrderingNonNative, 44
setBytesInChar, 45
setBytesInDouble, 45
setBytesInFloat, 45
setBytesInInteger, 45
setBytesInLong, 45
setBytesInLongDouble, 45
setBytesInShort, 45
setCharRepresentation, 45
setDoubleFloatRepresentation, 45
setExtraFloatRepresentation, 45
setIntegerRepresentation, 45
setSingleFloatRepresentation, 45
writeHeader, 45
writeNativeHeader, 43

DataTraits, 47
clear, 48
DataTraits, 48
getAtomSize, 48
getDimension, 48
getType, 48
getTypeName, 48
isArray, 48
isDefined, 48
isScalar, 48
isUndefined, 48
loadConfigurationFromFP, 48
operator!=, 49
operator=, 49
operator==, 49
printOn, 49
saveConfigurationToFP, 48
setDimension, 49
setTraits, 49
setType, 49

FieldDescriptor, 50
bitsToObj, 52
bytesNeeded, 52
cbitsToObj, 52
cbytesNeeded, 52
FieldDescriptor, 51, 52
getDimension, 52
getTraits, 52
getType, 52
objToBits, 52
objToCbits, 52
operator!=, 53
operator=, 53
operator==, 53
printOn, 53
setDimension, 52
setTraits, 52
setType, 53

FileStreamPipe, 54
FileStreamPipe, 55
~FileStreamPipe, 55
getFileDescriptor, 56
isFilePipe, 55
printOn, 56
successfulOpen, 56

InputFileStreamPipe, 57
_readPool, 58

bytesFree, 59
bytesReady, 59
createPipeReader, 59
get, 59
getByteOffset, 59
getFileSize, 59
getInputCharacteristics, 59
getInputType, 59
getSkippingPeek, 59
InputFileStreamPipe, 58
~InputFileStreamPipe, 59
isEmpty, 59
isFull, 59
isInputPipe, 59
peek, 60
printOn, 60
put, 60
setByteOffset, 60

Obj, 61
_setClassName, 62
abort, 62
error, 62
gentleWarning, 62
getClassName, 62
getObjectName, 62
hash, 62
info, 62
inspect, 62
notImplemented, 63
Obj, 62
~Obj, 62
printOn, 63
shouldNotImplement, 63
subclassResponsibility, 63
warning, 62

OutputFileStreamPipe, 64
_flushPool, 65
bytesFree, 66
bytesReady, 66
createPipeReader, 66
get, 66
getByteOffset, 66
getFileSize, 66
getSkippingPeek, 66
isEmpty, 66
isFull, 66
isOutputPipe, 66
OutputFileStreamPipe, 65
~OutputFileStreamPipe, 65
peek, 66
printOn, 67
put, 66
setByteOffset, 66
truncateAfterByte, 67

PipeReader, 68
getAttributes, 70
getData, 69
getDescriptor, 69
getFileSize, 70
getOffset, 70
getPacketDataPtr, 69
getPacketHeader, 70
PipeReader, 69
~PipeReader, 69
printOn, 70
rewind, 70
setOffset, 70
successfulConnection, 70

PipeWriter, 71
_writeSddfHeader, 72
addPipe, 72
deletePipe, 72
getFileSize, 73
getOffset, 73
PipeWriter, 72
~PipeWriter, 72
putAttributes, 72, 73
putCommand, 73
putData, 73
putDescriptor, 73
putPacket, 73
rewind, 73

setOffset, 73
successfulConnection, 73

RecordDictionary, 74
contains, 75
entryCount, 75
fetch, 75
getArrayP, 75, 76
getValue, 76
getValueP, 76
insert, 76
printOn, 76
RecordDictionary, 75
~RecordDictionary, 75
setValue, 76

RecordDictionaryPIterator, 77
first, 78
next, 78
RecordDictionaryPIterator, 78
~RecordDictionaryPIterator, 78

RecordDossier, 79
bitsToObj, 81
bytesNeeded, 81
cbitsToObj, 81
cbytesNeeded, 81
entryCount, 81
getArrayP, 81
getField, 81
getFieldID, 81
getTag, 81
getValue, 82
getValueP, 82
objToBits, 82
objToCbits, 82
operator=, 82
printOn, 82
RecordDossier, 80, 81
~RecordDossier, 81
setValue, 82

StreamPipe, 83
bytesFree, 84
bytesReady, 85
createPipeReader, 85
get, 85
getByteOffset, 85
getFileSize, 85
getInputCharacteristics, 85
getInputType, 85
getSkippingPeek, 85
isEmpty, 85
isFull, 85
peek, 85
printOn, 86
put, 85
setByteOffset, 86
StreamPipe, 84
~StreamPipe, 84
truncateAfterByte, 86

StructureDescriptor, 87
bitsToObj, 89
bytesNeeded, 89
cbitsToObj, 89
cbytesNeeded, 89
contains, 89
entryCount, 89
fetch, 89
insert, 90
isMatchFor, 90
objToBits, 90
objToCbits, 90
operator=, 90
printOn, 90
remove, 90
StructureDescriptor, 89
~StructureDescriptor, 89

StructureDescriptorIterator, 91
first, 92
next, 92
StructureDescriptorIterator, 92
~StructureDescriptorIterator, 92

Value, 93

Array*, 98
bitsToObj, 95
bytesNeeded, 95
cbitsToObj, 95
cbytesNeeded, 95
char, 98
double, 98
float, 98
getResultType, 96
getTraits, 96
getValueAs, 96
int, 98
isDefined, 96
isUndefined, 96
objToBits, 96
objToCbits, 96
operator, 97
operator!=, 97
operator+, 97
operator<, 97
operator<=, 97
operator=, 96, 97
operator==, 97
operator>, 97
operator>=, 97
operator*, 98
operator/, 98
printOn, 98
setQuantity, 96
Value, 94, 95
~Value, 95

Index to the Methods

_badInput
    AsciiPipeReader, 8
_checkBufferSize
    AsciiPipeWriter, 12
_flushPool
    OutputFileStreamPipe, 65
_getArray
    AsciiPipeReader, 8
_getAttributes
    AsciiPipeReader, 9
_getCommandTag
    AsciiPipeReader, 9
_getDataTag
    AsciiPipeReader, 9
_getDescriptorTag
    AsciiPipeReader, 9
_getField
    AsciiPipeReader, 9
_getPipeAttributes
    AsciiPipeReader, 9
_getString
    AsciiPipeReader, 9
_getType
    AsciiPipeReader, 9
_getValue
    AsciiPipeReader, 9
_increaseBufferSize
    AsciiPipeWriter, 12
_putArray
    AsciiPipeWriter, 12
_putC
    AsciiPipeWriter, 13
_putValue
    AsciiPipeWriter, 13
_readPacket
    AsciiPipeReader, 9
    BinaryPipeReader, 27
    ConversionPipeReader, 36
_readPool
    InputFileStreamPipe, 58
_setClassName
    Obj, 62
_setObjectName
    BasePipe, 24
_skipWhiteSpace
    AsciiPipeReader, 9
_writeCString
    AsciiPipeWriter, 13
_writePacket
    AsciiPipeWriter, 13
    BinaryPipeWriter, 30
    ConversionPipeWriter, 39
_writeSddfHeader
    AsciiPipeWriter, 13
    BinaryPipeWriter, 30
    ConversionPipeWriter, 39
    PipeWriter, 72

abort
    Obj, 62
addPipe
    AsciiPipeWriter, 13
    ConversionPipeWriter, 40
    PipeWriter, 72
Array
    Array, 3
~Array
    Array, 3
Array*
    Value, 98
AsciiPipeReader
    AsciiPipeReader, 10
~AsciiPipeReader
    AsciiPipeReader, 10
AsciiPipeWriter
    AsciiPipeWriter, 13
~AsciiPipeWriter
    AsciiPipeWriter, 13
atomSize
    Array, 3
Attributes
    Attributes, 16
~Attributes
    Attributes, 17

AttributesIterator
    AttributesIterator, 20
~AttributesIterator
    AttributesIterator, 20

BaseDescriptor
    BaseDescriptor, 22
~BaseDescriptor
    BaseDescriptor, 22
BasePipe
    BasePipe, 24
~BasePipe
    BasePipe, 24
BinaryPipeReader
    BinaryPipeReader, 27
~BinaryPipeReader
    BinaryPipeReader, 27
BinaryPipeWriter
    BinaryPipeWriter, 30
~BinaryPipeWriter
    BinaryPipeWriter, 30
bitsToObj
    Array, 3
    Attributes, 17
    FieldDescriptor, 52
    RecordDossier, 81
    StructureDescriptor, 89
    Value, 95
bytesFree
    InputFileStreamPipe, 59
    OutputFileStreamPipe, 66
    StreamPipe, 84
bytesNeeded
    Array, 3
    Attributes, 17
    FieldDescriptor, 52
    RecordDossier, 81
    StructureDescriptor, 89
    Value, 95
bytesReady
    InputFileStreamPipe, 59
    OutputFileStreamPipe, 66
    StreamPipe, 85
cbitsToObj
    Array, 3
    Attributes, 17
    FieldDescriptor, 52
    RecordDossier, 81
    StructureDescriptor, 89
    Value, 95
cbytesNeeded
    Array, 3
    Attributes, 17
    FieldDescriptor, 52
    RecordDossier, 81
    StructureDescriptor, 89
    Value, 95
char
    Value, 98
char*
    CString, 34
clear
    DataTraits, 48
clearDelta
    Array, 3
clearEntries
    Attributes, 17
concat
    CString, 33
contains
    Attributes, 17
    RecordDictionary, 75
    StructureDescriptor, 89
ConversionPipeReader
    ConversionPipeReader, 36
~ConversionPipeReader
    ConversionPipeReader, 36
ConversionPipeWriter
    ConversionPipeWriter, 39
~ConversionPipeWriter
    ConversionPipeWriter, 40
createPipeReader
    InputFileStreamPipe, 59
    OutputFileStreamPipe, 66
    StreamPipe, 85
CString
    CString, 33

~CString
CString, 33

DataCharacteristics
DataCharacteristics, 43
DataTraits
DataTraits, 48
deletePipe
PipeWriter, 72
double
Value, 98

entryCount
Attributes, 17
RecordDictionary, 75
RecordDossier, 81
StructureDescriptor, 89
error
Obj, 62

fetch
Attributes, 17
RecordDictionary, 75
StructureDescriptor, 89
FieldDescriptor
FieldDescriptor, 51, 52
FileStreamPipe
FileStreamPipe, 55
~FileStreamPipe
FileStreamPipe, 55
first
AttributesIterator, 20
RecordDictionaryPIterator, 78
StructureDescriptorIterator, 92
float
Value, 98
functionGet
DataCharacteristics, 43
functionPut
DataCharacteristics, 43, 44

gentleWarning
Obj, 62
get
InputFileStreamPipe, 59
OutputFileStreamPipe, 66
StreamPipe, 85
getArrayP
RecordDictionary, 75, 76
getAtomSize
DataCharacteristics, 44
DataTraits, 48
getAttributes
BaseDescriptor, 22
PipeReader, 70
getByteOffset
InputFileStreamPipe, 59
OutputFileStreamPipe, 66
StreamPipe, 85
getCellCount
Array, 3
getCellValue
Array, 3, 4
getClassName
Obj, 62
getData
AsciiPipeReader, 10
BinaryPipeReader, 27
ConversionPipeReader, 36
PipeReader, 69
getDeltaCount
Array, 4
getDeltaValue
Array, 4
getDescriptor
AsciiPipeReader, 10
BinaryPipeReader, 27
ConversionPipeReader, 36
PipeReader, 69
getDimension
Array, 4
DataTraits, 48
FieldDescriptor, 52
getDimSizes
Array, 4
getField
RecordDossier, 81
getFieldID

RecordDossier, 81
getFileDescriptor
FileStreamPipe, 56
getFileSize
InputFileStreamPipe, 59
OutputFileStreamPipe, 66
PipeReader, 70
PipeWriter, 73
StreamPipe, 85
getInputCharacteristics
InputFileStreamPipe, 59
StreamPipe, 85
getInputType
InputFileStreamPipe, 59
StreamPipe, 85
getInputWrapper
BasePipe, 24
getName
BaseDescriptor, 22
getObjectName
BaseDescriptor, 22
BasePipe, 24
Obj, 62
getOffset
PipeReader, 70
PipeWriter, 73
getOutputWrapper
BasePipe, 24
getPacketDataPtr
AsciiPipeReader, 10
BinaryPipeReader, 27
ConversionPipeReader, 37
PipeReader, 69
getPacketHeader
AsciiPipeReader, 10
BinaryPipeReader, 28
ConversionPipeReader, 37
PipeReader, 70
getResultType
Value, 96
getSkippingPeek
InputFileStreamPipe, 59
OutputFileStreamPipe, 66
StreamPipe, 85
getTag
RecordDossier, 81
getTheCellValue
Array, 4
getTraits
FieldDescriptor, 52
Value, 96
getType
Array, 4
DataTraits, 48
FieldDescriptor, 52
getTypeName
DataTraits, 48
getValue
CString, 33
RecordDictionary, 76
RecordDossier, 82
getValueAs
Value, 96
getValueP
RecordDictionary, 76
RecordDossier, 82

hash
Obj, 62

info
Obj, 62
initializeFromFile
DataCharacteristics, 44
InputFileStreamPipe
InputFileStreamPipe, 58
~InputFileStreamPipe
InputFileStreamPipe, 59
insert
Attributes, 17
RecordDictionary, 76
StructureDescriptor, 90
insertAttributes
BaseDescriptor, 22
inspect
Obj, 62
int
Value, 98

intSize
DataCharacteristics, 44
isArray
DataTraits, 48
isDefined
DataTraits, 48
Value, 96
isDeltaArray
Array, 5
isEmpty
InputFileStreamPipe, 59
OutputFileStreamPipe, 66
StreamPipe, 85
isFilePipe
BasePipe, 24
FileStreamPipe, 55
isFull
InputFileStreamPipe, 59
OutputFileStreamPipe, 66
StreamPipe, 85
isInputPipe
BasePipe, 24
InputFileStreamPipe, 59
isMatchFor
StructureDescriptor, 90
isOutputPipe
BasePipe, 24
OutputFileStreamPipe, 66
isScalar
DataTraits, 48
isUndefined
DataTraits, 48
Value, 96

length
CString, 33
loadConfigurationFromFP
DataTraits, 48

next
AttributesIterator, 20
RecordDictionaryPIterator, 78
StructureDescriptorIterator, 92
notImplemented
Obj, 63
notNative
DataCharacteristics, 44
notNull
BaseDescriptor, 22

Obj
Obj, 62
~Obj
Obj, 62
objToBits
Array, 5
Attributes, 17
FieldDescriptor, 52
RecordDossier, 82
StructureDescriptor, 90
Value, 96
objToCbits
Array, 5
Attributes, 18
FieldDescriptor, 52
RecordDossier, 82
StructureDescriptor, 90
Value, 96
operator!=
Attributes, 18
CString, 34
DataTraits, 49
FieldDescriptor, 53
Value, 97
operator+
CString, 34
Value, 97
operator<
Value, 97
operator<=
CString, 34
Value, 97
operator=
Array, 6
Attributes, 18
CString, 33
DataTraits, 49
FieldDescriptor, 53

RecordDossier, 82
StructureDescriptor, 90
Value, 96, 97
operator==
Attributes, 18
DataTraits, 49
FieldDescriptor, 53
Value, 97
operator>
Value, 97
operator>=
CString, 34
Value, 97
operator*
Value, 98
operator/
Value, 98
operator[]
Attributes, 18
CString, 34
OutputFileStreamPipe
OutputFileStreamPipe, 65
~OutputFileStreamPipe
OutputFileStreamPipe, 65

peek
InputFileStreamPipe, 60
OutputFileStreamPipe, 66
StreamPipe, 85
PipeReader
PipeReader, 69
~PipeReader
PipeReader, 69
PipeWriter
PipeWriter, 72
~PipeWriter
PipeWriter, 72
printOn
Array, 6
Attributes, 18
BaseDescriptor, 22
BasePipe, 25
CString, 34
DataTraits, 49
FieldDescriptor, 53
FileStreamPipe, 56
InputFileStreamPipe, 60
Obj, 63
OutputFileStreamPipe, 67
PipeReader, 70
RecordDictionary, 76
RecordDossier, 82
StreamPipe, 86
StructureDescriptor, 90
Value, 98
put
InputFileStreamPipe, 60
OutputFileStreamPipe, 66
StreamPipe, 85
putAttributes
AsciiPipeWriter, 13
BinaryPipeWriter, 30, 31
ConversionPipeWriter, 40
PipeWriter, 72, 73
putCommand
AsciiPipeWriter, 13
BinaryPipeWriter, 31
ConversionPipeWriter, 40
PipeWriter, 73
putData
AsciiPipeWriter, 13
BinaryPipeWriter, 31
ConversionPipeWriter, 40
PipeWriter, 73
putDescriptor
AsciiPipeWriter, 14
BinaryPipeWriter, 31
ConversionPipeWriter, 40
PipeWriter, 73
putPacket
AsciiPipeWriter, 14
BinaryPipeWriter, 31
ConversionPipeWriter, 40
PipeWriter, 73

RecordDictionary
RecordDictionary, 75
~RecordDictionary

RecordDictionary, 75
RecordDictionaryPIterator
RecordDictionaryPIterator, 78
~RecordDictionaryPIterator
RecordDictionaryPIterator, 78
RecordDossier
RecordDossier, 80, 81
~RecordDossier
RecordDossier, 81
remove
Attributes, 18
StructureDescriptor, 90
resetDelta
Array, 5
resetIterateCount
Array, 5
rewind
PipeReader, 70
PipeWriter, 73

saveConfigurationToFP
DataTraits, 48
setAttributes
BaseDescriptor, 22
setByteOffset
InputFileStreamPipe, 60
OutputFileStreamPipe, 66
StreamPipe, 86
setByteOrdering
DataCharacteristics, 44
setByteOrderingNonNative
DataCharacteristics, 44
setBytesInChar
DataCharacteristics, 45
setBytesInDouble
DataCharacteristics, 45
setBytesInFloat
DataCharacteristics, 45
setBytesInInteger
DataCharacteristics, 45
setBytesInLong
DataCharacteristics, 45
setBytesInLongDouble
DataCharacteristics, 45
setBytesInShort
DataCharacteristics, 45
setCellString
Array, 5
setCellValue
Array, 5
setCellValues
Array, 5
setCharRepresentation
DataCharacteristics, 45
setDeltaIndex
Array, 6
setDimension
DataTraits, 49
FieldDescriptor, 52
setDimSizes
Array, 5
setDoubleFloatRepresentation
DataCharacteristics, 45
setExtraFloatRepresentation
DataCharacteristics, 45
setInputWrapper
BasePipe, 24
setIntegerRepresentation
DataCharacteristics, 45
setName
BaseDescriptor, 22
setOffset
PipeReader, 70
PipeWriter, 73
setOutputWrapper
BasePipe, 25
setQuantity
Value, 96
setSingleFloatRepresentation
DataCharacteristics, 45
setTheCellValue
Array, 5
setTraits
DataTraits, 49
FieldDescriptor, 52
setType
DataTraits, 49
FieldDescriptor, 53

setValue
    RecordDictionary, 76
    RecordDossier, 82
shouldNotImplement
    Obj, 63
StreamPipe
    StreamPipe, 84
~StreamPipe
    StreamPipe, 84
StructureDescriptor
    StructureDescriptor, 89
~StructureDescriptor
    StructureDescriptor, 89
StructureDescriptorIterator
    StructureDescriptorIterator, 92
~StructureDescriptorIterator
    StructureDescriptorIterator, 92
subclassResponsibility
    Obj, 63
successfulConnection
    PipeReader, 70
    PipeWriter, 73
successfulOpen
    FileStreamPipe, 56

theValue
    AttributesIterator, 20
truncateAfterByte
    OutputFileStreamPipe, 67
    StreamPipe, 86

Value
    Value, 94, 95
~Value
    Value, 95

warning
    Obj, 62
writeHeader
    DataCharacteristics, 45
writeNativeHeader
    DataCharacteristics, 43