HP Vertica Java SDK Documentation Version 7.0

Tue Feb 11 2014

Contents

Hierarchical Index

Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Class Index

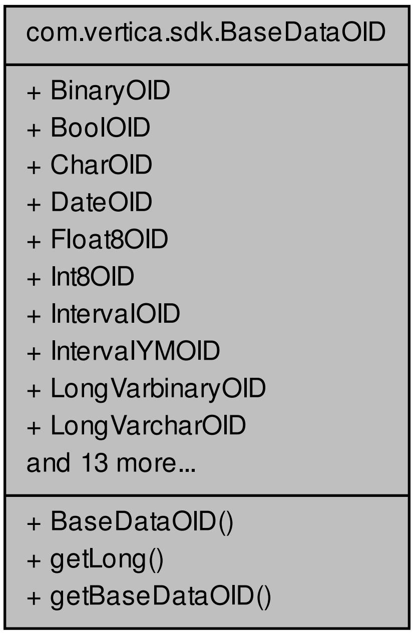
Class List

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

Class Documentation

com.vertica.sdk.BaseDataOID Enum Reference

Collaboration diagram for com.vertica.sdk.BaseDataOID:



Public Member Functions

- **BaseDataOID** (long oid)
- long **getLong** ()

Static Public Member Functions

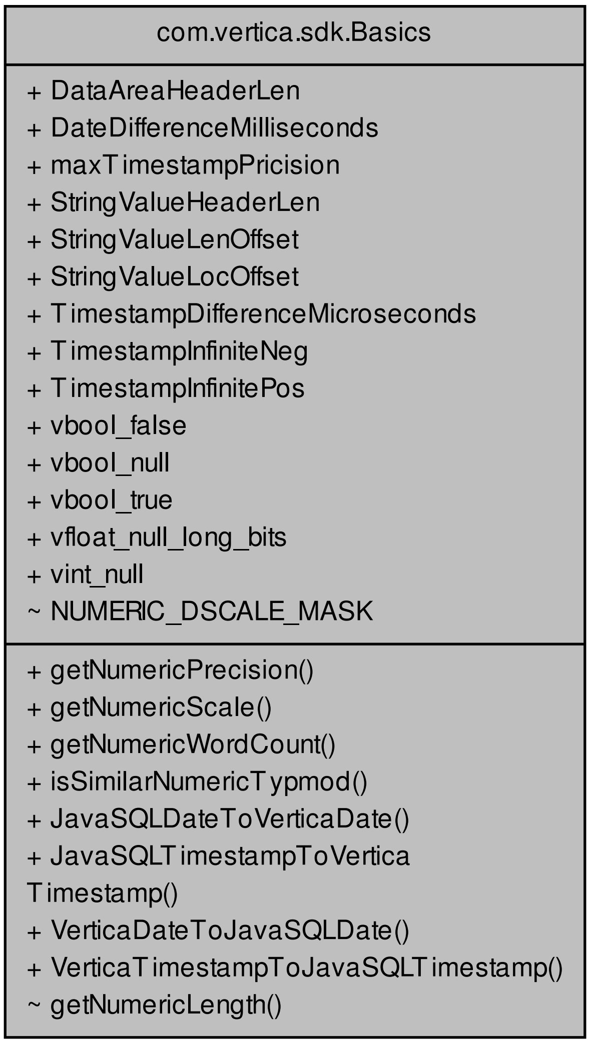
- static BaseDataOID **getBaseDataOID** (long oid)

Public Attributes

- **BinaryOID** =(117)
- **BoolOID** =(5)
- **CharOID** =(8)
- **DateOID** =(10)
- **Float8OID** =(7)
- **Int8OID** =(6)
- **IntervalOID** =(14)
- **IntervalYMOID** =(114)
- **LongVarbinaryOID** =(116)
- **LongVarcharOID** =(115)
- **NumericOID** =(16)
- **RecordOID** =(3)
- **RLETuple** =(18)
- **TimeOID** =(11)
- **TimestampOID** =(12)
- **TimestampTzOID** =(13)
- **TimeTzOID** =(15)
- **UnknownOID** =(4)
- **VarbinaryOID** =(17)
- **VarcharOID** =(9)
- **VPosOID** =(2)
- **VTuple** =(1)
- **VUnspecOID** =(0)

com.vertica.sdk.Basics Class Reference

Collaboration diagram for com.vertica.sdk.Basics:



Static Public Member Functions

- static int getNumericPrecision (int typmod)

  Get Numeric precision from typmod.
- static int getNumericScale (int typmod)

  Get Numeric scale from typmod.
- static int getNumericWordCount (int precision)

  Get Numeric word count from precision.
- static boolean isSimilarNumericTypmod (int a, int b)

  Return true if these have the same EE representation.
- static long **JavaSQLDateToVerticaDate** (java.sql.Date d)

- static long **JavaSQLTimestampToVerticaTimestamp** (java.sql.Timestamp ts)
- static java.sql.Date VerticaDateToJavaSQLDate (long num_days)
- static java.sql.Timestamp VerticaTimestampToJavaSQLTimestamp (long vts)

Static Public Attributes

- static final int **DataAreaHeaderLen** = 16
- static final long **DateDifferenceMilliseconds** = java.sql.Timestamp.valueOf("2000-01-01 00:00:00").getTime()
- static final int **maxTimestampPricision** = 6
- static final int **StringValueHeaderLen** = 8
- static final int **StringValueLenOffset** = 0
- static final int **StringValueLocOffset** = 4
- static final long **TimestampDifferenceMicroseconds** = 1000 ∗ java.sql.Timestamp.valueOf("2000-01-01 00-:00:00.000000000").getTime()
- static final long **TimestampInfiniteNeg** = -0x7fffffffffffffffL
- static final long **TimestampInfinitePos** = 0x7fffffffffffffffL
- static final byte **vbool_false** = 0
- static final byte **vbool_null** = 2
- static final byte **vbool_true** = 1
- static final long **vfloat_null_long_bits** = 0x7ffffffffffffffeL
- static final long **vint_null** = 0x8000000000000000L

Member Function Documentation

static java.sql.Date com.vertica.sdk.Basics.VerticaDateToJavaSQLDate (long *num_days*) [static]

Parameters

num_days	number of days since 2000-01-01

static java.sql.Timestamp com.vertica.sdk.Basics.VerticaTimestampToJavaSQLTimestamp (long *vts*) [static]

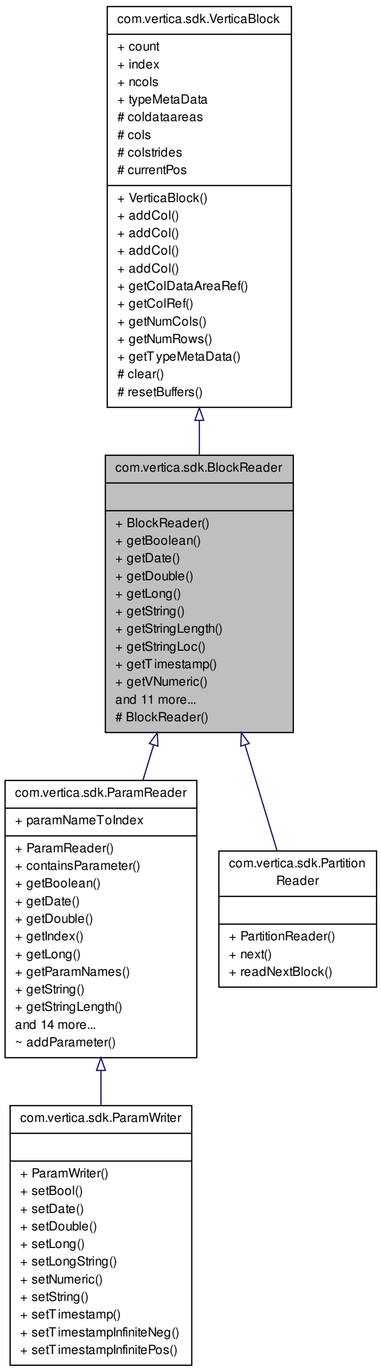
Parameters

vts	number of microseconds since 2000-01-01 00:00:00 GMT

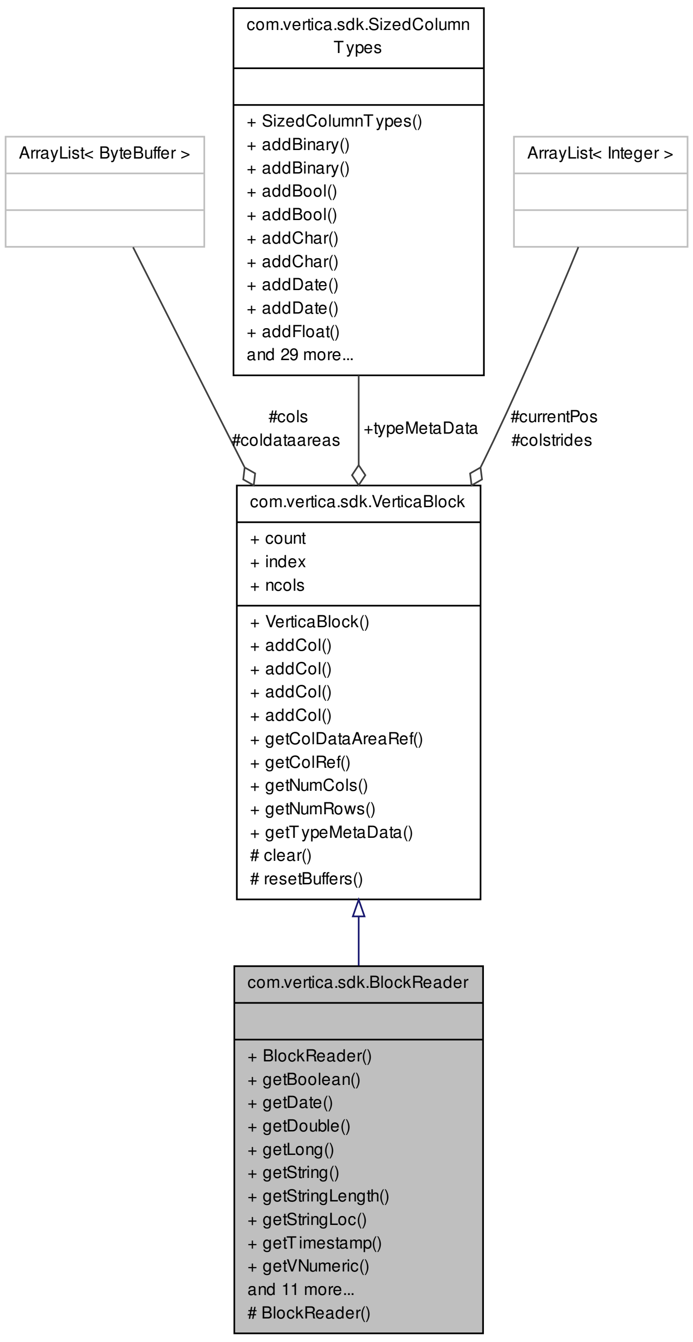
com.vertica.sdk.BlockReader Class Reference

Iterator interface for reading rows in a Vertica block.

Inheritance diagram for com.vertica.sdk.BlockReader:



Collaboration diagram for com.vertica.sdk.BlockReader:



Public Member Functions

- void addCol (ByteBuffer arg, int colstride, VerticaType dt, String colName)
- void addCol (ByteBuffer arg, int colstride, VerticaType dt)
- void addCol (ByteBuffer arg, ByteBuffer da, int colstride, VerticaType dt)
- void addCol (ByteBuffer arg, ByteBuffer da, int colstride, VerticaType dt, String colName)
- boolean getBoolean (int idx)

Get a BOOLEAN value from the input row.
- ByteBuffer getColDataAreaRef (int idx)
- ByteBuffer getColRef (int idx)
- java.sql.Date getDate (int idx)

  Get a DATE value from the input row.
- double getDouble (int idx)

  Get a DOUBLE value from the input row.
- long getLong (int idx)

  Get a LONG INTEGER value from the input row.
- int getNumCols ()
- int getNumRows ()
- String getString (int idx)

  Get a reference to an VARCHAR/CHAR/VARBINARY/BINARY value from the input row.
- int getStringLength (int idx)

  Get length of the String from the input row.
- int getStringLoc (int idx)

  Get 'location' of the String from the input row.
- java.sql.Timestamp getTimestamp (int idx)

  Get a TIMESTAMP value from the input row.
- SizedColumnTypes getTypeMetaData ()
- VNumeric getVNumeric (int idx)

  Get a reference to a VNumeric value from the input row.
- VString getVString (int idx)

  Get a reference from the input row to an VString value, which represents a SQL VARCHAR/CHAR/VARBINARY/BINARY value.
- boolean isBooleanNull (int idx)

  Check whether a value from the input row is NULL in BOOLEAN type.
- boolean isDateNull (int idx)

  Check whether a value from the input row is NULL in DATE type.
- boolean isDoubleNull (int idx)

  Check whether a value from the input row is NULL in DOUBLE type.
- boolean isLongNull (int idx)

  Check whether a value from the input row is NULL in LONG INTERGER type.
- boolean isStringNull (int idx)

  Check whether a value from the input row is NULL in SQL VARCHAR/CHAR/VARBINARY/BINARY type.
- boolean isTimestampInfinite (int idx)

  Check whether a TIMESTAMP value from the input row represents 'infinity'.
- boolean isTimestampInfiniteNeg (int idx)

  Check whether a TIMESTAMP value from the input row represents '-infinity'.
- boolean isTimestampInfinitePos (int idx)

  Check whether a TIMESTAMP value from the input row represents '+infinity'.
- boolean isTimestampNull (int idx)

  Check whether a value from the input row is NULL in TIMESTAMP type.
- boolean next () throws UdfException, DestroyInvocation

Public Attributes

- int **count**
- int **index**
- int **ncols**
- SizedColumnTypes **typeMetaData**

Protected Member Functions

- **BlockReader** (int _ncols, int _rowcount)
- void **clear** ()
- void **resetBuffers** ()

Protected Attributes

- ArrayList< ByteBuffer > **coldataareas**
- ArrayList< ByteBuffer > **cols**
- ArrayList< Integer > **colstrides**
- ArrayList< Integer > **currentPos**

Detailed Description

Iterator interface for reading rows in a Vertica block.

This class provides the input to the ScalarFunction.processBlock() function. You extract values from the input row using data type specific functions to extract each column value. You can also determine the number of columns and their data types, if your processBlock function does not have hard-coded input expectations.

Member Function Documentation

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, int *colstride*, VerticaType *dt*, String *colName*) `[inherited]`

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
colstride	The stride between data instances.
dt	The type of input.
colName	Name of the column

Referenced by com.vertica.sdk.VerticaBlock.addCol().

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, int *colstride*, VerticaType *dt*) `[inherited]`

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
colstride	The stride between data instances.
dt	The type of input.

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, ByteBuffer *da*, int *colstride*, VerticaType *dt*) `[inherited]`

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
da	The location to find out of band string data.
colstride	The stride between data instances.
dt	The type of input.

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, ByteBuffer *da*, int *colstride*, VerticaType *dt*, String *colName*) `[inherited]`

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
da	The location to find out of band string data.
colstride	The stride between data instances.
dt	The type of input.
colName	Name of the column

boolean com.vertica.sdk.BlockReader.getBoolean (int *idx*)

Get a BOOLEAN value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The value of the idx'th argument, cast as a BOOLEAN.

ByteBuffer com.vertica.sdk.VerticaBlock.getColDataAreaRef (int *idx*) `[inherited]`

Get the ByteBuffer that stores out of line string data (Data Area) for the idx'th argument

Parameters

idx	

Returns

Referenced by com.vertica.sdk.BlockReader.getVString().

ByteBuffer com.vertica.sdk.VerticaBlock.getColRef (int *idx*) `[inherited]`

Returns

a ByteBuffer to the idx'th argument, containing data for the column

Example:

```
* ByteBuffer a = arg_reader.getColPtr(0);
*
```

Referenced by com.vertica.sdk.PartitionWriter.copyFromInput(), com.vertica.sdk.BlockReader.getBoolean(), com.vertica.sdk.BlockReader.getDouble(), com.vertica.sdk.BlockReader.getLong(), com.vertica.sdk.BlockReader.getStringLength(), com.vertica.sdk.BlockReader.getStringLoc(), com.vertica.sdk.BlockReader.getVNumeric(), com.vertica.sdk.BlockReader.getVString(), com.vertica.sdk.BlockWriter.getVStringWriter(), com.vertica.sdk.BlockReader.isBooleanNull(), com.vertica.sdk.ParamWriter.setBool(), com.vertica.sdk.BlockWriter.setBoolean(), com.vertica.sdk.BlockWriter.setBooleanNull(), com.vertica.sdk.ParamWriter.setDouble(), com.vertica.sdk.BlockWriter.setDouble(), com.vertica.sdk.BlockWriter.setDoubleNull(), com.vertica.sdk.PartitionWriter.setLong(), com.vertica.sdk.BlockWriter.setLongNull(), com.vertica.sdk.ParamWriter.setLongString(), com.vertica.sdk.BlockWriter.setNumeric(), com.vertica.sdk.ParamWriter.setNumeric(), com.vertica.sdk.BlockWriter.setString(), com.vertica.sdk.ParamWriter.setString(), and com.vertica.sdk.BlockWriter.setStringNull().

java.sql.Date com.vertica.sdk.BlockReader.getDate (int *idx*)

Get a DATE value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The value of the idx'th argument, cast as a DATE; null if the column is NULL.

double com.vertica.sdk.BlockReader.getDouble (int *idx*)

Get a DOUBLE value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The value of the idx'th argument, cast as a DOUBLE.

long com.vertica.sdk.BlockReader.getLong (int *idx*)

Get a LONG INTEGER value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The value of the idx'th argument, cast as a LONG INTEGER.

Example:

```
* long a = arg_reader.getLong(0);
*
```

Referenced by com.vertica.sdk.BlockReader.getDate(), com.vertica.sdk.BlockReader.getTimestamp(), com.vertica.sdk.BlockReader.isDoubleNull(), com.vertica.sdk.BlockReader.isLongNull(), com.vertica.sdk.BlockReader.isTimestampInfiniteNeg(), and com.vertica.sdk.BlockReader.isTimestampInfinitePos().

int com.vertica.sdk.VerticaBlock.getNumCols ( ) `[inherited]`

Returns

the number of arguments held by this reader.

int com.vertica.sdk.VerticaBlock.getNumRows ( ) `[inherited]`

Returns

the number of rows held by this block.

String com.vertica.sdk.BlockReader.getString (int *idx*)

Get a reference to an VARCHAR/CHAR/VARBINARY/BINARY value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

a reference to the idx'th argument, cast as an String.

int com.vertica.sdk.BlockReader.getStringLength (int *idx*)

Get length of the String from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The length of the String in specified column.

Referenced by com.vertica.sdk.BlockReader.getVString(), and com.vertica.sdk.BlockReader.isStringNull().

int com.vertica.sdk.BlockReader.getStringLoc (int *idx*)

Get 'location' of the String from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The location of the String in specified column. If zero, data is inlined immediately after the header, otherwise data is at offset loc within the data area.

Referenced by com.vertica.sdk.BlockReader.getVString().

java.sql.Timestamp com.vertica.sdk.BlockReader.getTimestamp (int *idx*)

Get a TIMESTAMP value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The value of the idx'th argument, cast as a TIMESTAMP; null if the column is NULL or represents 'infinity'.

SizedColumnTypes com.vertica.sdk.VerticaBlock.getTypeMetaData ( ) [inherited]

Returns

information about the types and numbers of arguments

Referenced by com.vertica.sdk.ParamReader.getType().

VNumeric com.vertica.sdk.BlockReader.getVNumeric (int *idx*)

Get a reference to a VNumeric value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

A reference to the retrieved value cast as a Numeric.

VString com.vertica.sdk.BlockReader.getVString (int *idx*)

Get a reference from the input row to an VString value, which represents a SQL VARCHAR/CHAR/VARBINARY/B-INARY value.

Parameters

idx	The column number to retrieve from the input row.

Returns

a reference to the idx'th argument, cast as an VString.

Referenced by com.vertica.sdk.BlockReader.getString().

boolean com.vertica.sdk.BlockReader.isBooleanNull (int *idx*)

Check whether a value from the input row is NULL in BOOLEAN type.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the value is NULL, false otherwise.

boolean com.vertica.sdk.BlockReader.isDateNull (int *idx*)

Check whether a value from the input row is NULL in DATE type.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the value is NULL, false otherwise.

Referenced by com.vertica.sdk.BlockReader.getDate().

boolean com.vertica.sdk.BlockReader.isDoubleNull (int *idx*)

Check whether a value from the input row is NULL in DOUBLE type.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the value is NULL, false otherwise.

boolean com.vertica.sdk.BlockReader.isLongNull (int *idx*)

Check whether a value from the input row is NULL in LONG INTERGER type.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the value is NULL, false otherwise.

Referenced by com.vertica.sdk.BlockReader.isDateNull(), and com.vertica.sdk.BlockReader.isTimestampNull().

boolean com.vertica.sdk.BlockReader.isStringNull (int *idx*)

Check whether a value from the input row is NULL in SQL VARCHAR/CHAR/VARBINARY/BINARY type.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the value is NULL, false otherwise.

Referenced by com.vertica.sdk.BlockReader.getString().

boolean com.vertica.sdk.BlockReader.isTimestampInfinite (int *idx*)

Check whether a TIMESTAMP value from the input row represents ’infinity’.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the TIMESTAMP value is '+infinity' or '-infinity', false otherwise.

Referenced by com.vertica.sdk.BlockReader.getTimestamp().

boolean com.vertica.sdk.BlockReader.isTimestampInfiniteNeg (int *idx*)

Check whether a TIMESTAMP value from the input row represents '-infinity'.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the TIMESTAMP value is '-infinity', false otherwise.

Referenced by com.vertica.sdk.BlockReader.isTimestampInfinite().

boolean com.vertica.sdk.BlockReader.isTimestampInfinitePos (int *idx*)

Check whether a TIMESTAMP value from the input row represents '+infinity'.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the TIMESTAMP value is '+infinity', false otherwise.

Referenced by com.vertica.sdk.BlockReader.isTimestampInfinite().

boolean com.vertica.sdk.BlockReader.isTimestampNull (int *idx*)

Check whether a value from the input row is NULL in TIMESTAMP type.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the value is NULL, false otherwise.

Referenced by com.vertica.sdk.BlockReader.getTimestamp().

boolean com.vertica.sdk.BlockReader.next ( ) throws UdfException, DestroyInvocation

Advance to the next record.

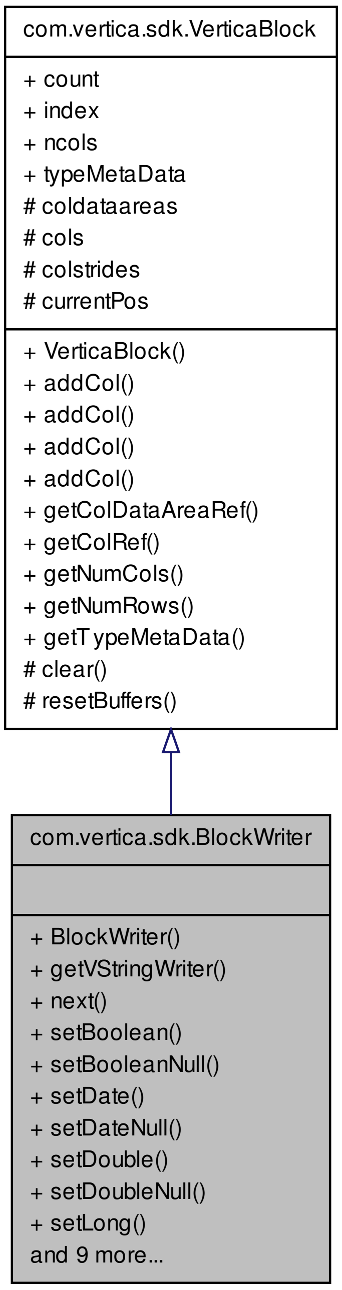
Returns

true if there are more rows to read, false otherwise.

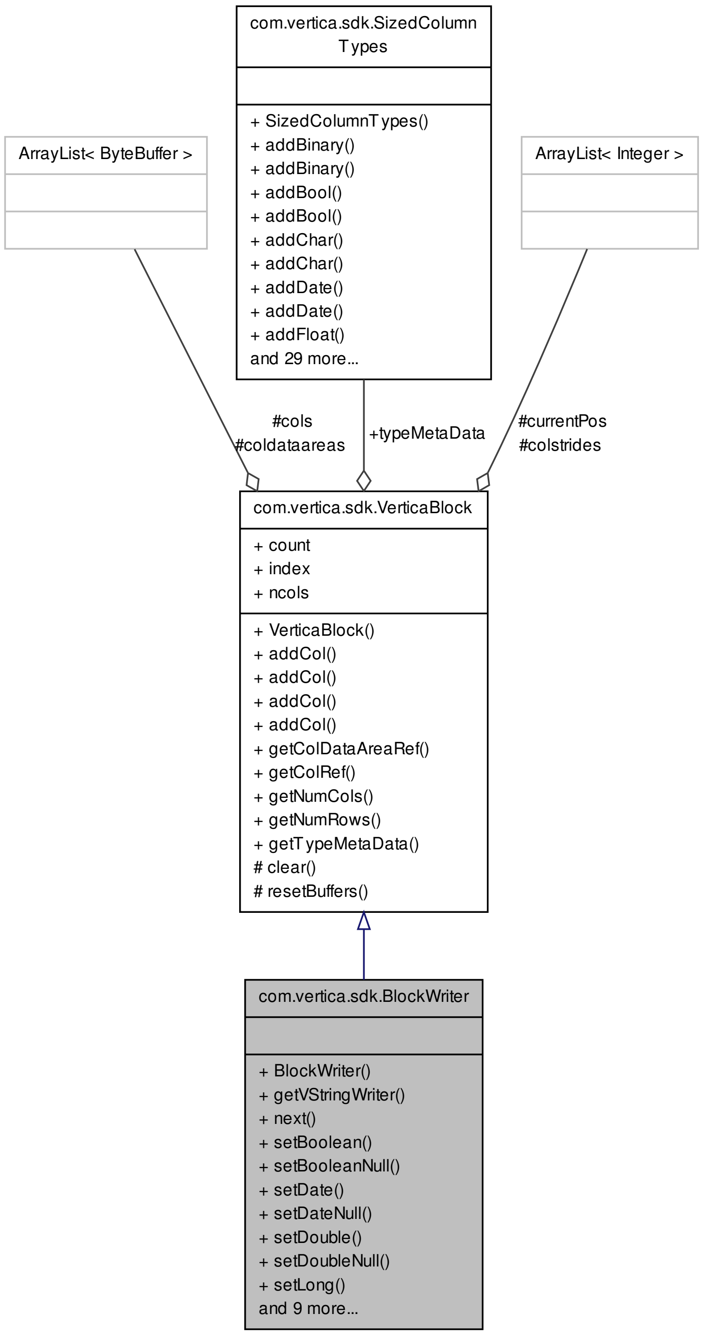
com.vertica.sdk.BlockWriter Class Reference

Iterator interface for writing rows to a Vertica block.

Inheritance diagram for com.vertica.sdk.BlockWriter:



Collaboration diagram for com.vertica.sdk.BlockWriter:



Public Member Functions

- **BlockWriter** (int rowcount, VerticaType returnType)
- void addCol (ByteBuffer arg, int colstride, VerticaType dt, String colName)
- void addCol (ByteBuffer arg, int colstride, VerticaType dt)
- void addCol (ByteBuffer arg, ByteBuffer da, int colstride, VerticaType dt)
- void addCol (ByteBuffer arg, ByteBuffer da, int colstride, VerticaType dt, String colName)

- ByteBuffer getColDataAreaRef (int idx)
- ByteBuffer getColRef (int idx)
- int getNumCols ()
- int getNumRows ()
- SizedColumnTypes getTypeMetaData ()
- VString getVStringWriter ()

  Allocates a new VString object to use as output.
- void next ()

  Complete writing this row of output and move to the next row.
- void setBoolean (boolean r)

  Adds a BOOLEAN value to the output row.
- void setBooleanNull ()

  Adds NULL as a BOOLEAN value to the output row.
- void setDate (java.sql.Date r)

  Adds a DATE value to the output row.
- void setDateNull ()

  Adds NULL as a DATE value to the output row.
- void setDouble (double r)

  Adds a DOUBLE value to the output row.
- void setDoubleNull ()

  Adds NULL as a DOUBLE value to the output row.
- void setLong (long r)

  Adds a LONG INTEGER value to the output row.
- void setLongNull ()

  Adds NULL as a LONG INTEGER value to the output row.
- void setNumeric (BigDecimal bd)

  Allocate a new VNumeric object to use as output.
- void **setNumericNull** ()
- void setString (String r)

  Adds a String value to the output row.
- void setStringNull ()

  Adds NULL as a String value to the output row.
- void setTimestamp (java.sql.Timestamp r)

  Adds a TIMESTAMP value to the output row.
- void setTimestampInfiniteNeg ()

  Adds a '-infinity' TIMESTAMP value to the output row.
- void setTimestampInfinitePos ()

  Adds a '+infinity' TIMESTAMP value to the output row.
- void setTimestampNull ()

  Adds NULL as a TIMESTAMP value to the output row.

Public Attributes

- int **count**
- int **index**
- int **ncols**
- SizedColumnTypes **typeMetaData**

Protected Member Functions

- void **clear** ()
- void **resetBuffers** ()

Protected Attributes

- ArrayList< ByteBuffer > **coldataareas**
- ArrayList< ByteBuffer > **cols**
- ArrayList< Integer > **colstrides**
- ArrayList< Integer > **currentPos**

Detailed Description

Iterator interface for writing rows to a Vertica block.

This class provides the output rows that ScalarFunction.processBlock() writes to.

Member Function Documentation

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, int *colstride*, VerticaType *dt*, String *colName*) `[inherited]`

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
colstride	The stride between data instances.
dt	The type of input.
colName	Name of the column

Referenced by com.vertica.sdk.VerticaBlock.addCol().

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, int *colstride*, VerticaType *dt*) `[inherited]`

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
colstride	The stride between data instances.
dt	The type of input.

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, ByteBuffer *da*, int *colstride*, VerticaType *dt*) `[inherited]`

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
da	The location to find out of band string data.
colstride	The stride between data instances.
dt	The type of input.

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, ByteBuffer *da*, int *colstride*, VerticaType *dt*, String *colName*) `[inherited]`

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
da	The location to find out of band string data.
colstride	The stride between data instances.
dt	The type of input.
colName	Name of the column

ByteBuffer com.vertica.sdk.VerticaBlock.getColDataAreaRef (int *idx*) `[inherited]`

Get the ByteBuffer that stores out of line string data (Data Area) for the idx'th argument

Parameters

idx	

Returns

Referenced by com.vertica.sdk.BlockReader.getVString().

ByteBuffer com.vertica.sdk.VerticaBlock.getColRef (int *idx*) `[inherited]`

Returns

a ByteBuffer to the idx'th argument, containing data for the column

Example:

```
* ByteBuffer a = arg_reader.getColPtr(0);
*
```

Referenced by com.vertica.sdk.PartitionWriter.copyFromInput(), com.vertica.sdk.BlockReader.getBoolean(), com.-vertica.sdk.BlockReader.getDouble(), com.vertica.sdk.BlockReader.getLong(), com.vertica.sdk.BlockReader.-getStringLength(), com.vertica.sdk.BlockReader.getStringLoc(), com.vertica.sdk.BlockReader.getVNumeric(), com.vertica.sdk.BlockReader.getVString(), com.vertica.sdk.BlockWriter.getVStringWriter(), com.vertica.sdk.-BlockReader.isBooleanNull(), com.vertica.sdk.ParamWriter.setBool(), com.vertica.sdk.BlockWriter.setBoolean(), com.vertica.sdk.BlockWriter.setBooleanNull(), com.vertica.sdk.ParamWriter.setDouble(), com.vertica.sdk.Block-Writer.setDouble(), com.vertica.sdk.BlockWriter.setDoubleNull(), com.vertica.sdk.PartitionWriter.setLong(), com.-vertica.sdk.BlockWriter.setLongNull(), com.vertica.sdk.ParamWriter.setLongString(), com.vertica.sdk.BlockWriter.-setNumeric(), com.vertica.sdk.ParamWriter.setNumeric(), com.vertica.sdk.BlockWriter.setString(), com.vertica.-sdk.ParamWriter.setString(), and com.vertica.sdk.BlockWriter.setStringNull().

int com.vertica.sdk.VerticaBlock.getNumCols ( ) `[inherited]`

Returns

the number of arguments held by this reader.

int com.vertica.sdk.VerticaBlock.getNumRows ( ) `[inherited]`

Returns

the number of rows held by this block.

SizedColumnTypes com.vertica.sdk.VerticaBlock.getTypeMetaData ( ) [inherited]

Returns

information about the types and numbers of arguments

Referenced by com.vertica.sdk.ParamReader.getType().

VString com.vertica.sdk.BlockWriter.getVStringWriter ( )

Allocates a new VString object to use as output.

Returns

A new VString object to hold output. This object automatically added to the output row.

void com.vertica.sdk.BlockWriter.next ( )

Complete writing this row of output and move to the next row.

void com.vertica.sdk.BlockWriter.setBoolean (boolean *r*)

Adds a BOOLEAN value to the output row.

Parameters

r	The BOOLEAN value to insert into the output row.

void com.vertica.sdk.BlockWriter.setDate (java.sql.Date *r*)

Adds a DATE value to the output row.

Parameters

r	The DATE value to insert into the output row.

void com.vertica.sdk.BlockWriter.setDouble (double *r*)

Adds a DOUBLE value to the output row.

Parameters

r	The DOUBLE value to insert into the output row.

void com.vertica.sdk.BlockWriter.setLong (long *r*)

Adds a LONG INTEGER value to the output row.

Setter methods

Parameters

r	The LONG INTEGER value to insert into the output row.

Referenced by com.vertica.sdk.BlockWriter.setDate(), com.vertica.sdk.BlockWriter.setTimestamp(), com.vertica.-sdk.BlockWriter.setTimestampInfiniteNeg(), and com.vertica.sdk.BlockWriter.setTimestampInfinitePos().

void com.vertica.sdk.BlockWriter.setNumeric (BigDecimal *bd*)

Allocate a new VNumeric object to use as output.

Returns

A new VNumeric object to hold output. This object automatically added to the output row.

void com.vertica.sdk.BlockWriter.setString (String *r*)

Adds a String value to the output row.

Parameters

r	The String value to insert into the output row.

void com.vertica.sdk.BlockWriter.setTimestamp (java.sql.Timestamp *r*)

Adds a TIMESTAMP value to the output row.

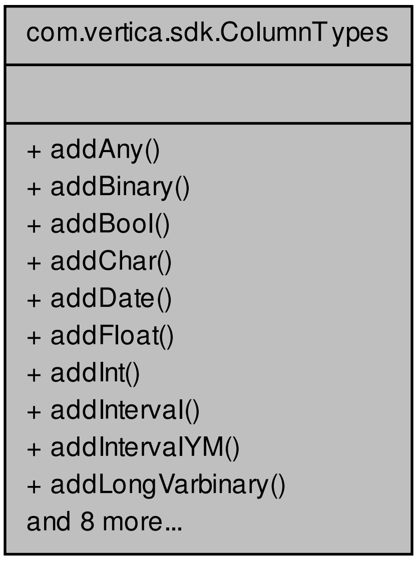
Parameters

r	The TIMESTAMP value to insert into the output row.

com.vertica.sdk.ColumnTypes Class Reference

Represents (unsized) types of the columns used as input/output of a User Defined Function/Transform Function.

Collaboration diagram for com.vertica.sdk.ColumnTypes:



Public Member Functions

- void addAny ()

  Indicates that function can take any number and type of arguments.
- void addBinary ()

  Adds a column of type BINARY.
- void addBool ()

  Adds a column of type BOOLEAN.
- void addChar ()

  Adds a column of type CHAR.
- void addDate ()

  Adds a column of type DATE.
- void addFloat ()

  Adds a column of type FLOAT.
- void addInt ()

  Adds a column of type INTEGER.
- void addInterval ()

  Adds a column of type INTERVAL/INTERVAL DAY TO SECOND.
- void addIntervalYM ()

  Adds a column of type INTERVAL YEAR TO MONTH.
- void addLongVarbinary ()

  Adds a column of type LONGVARBINARY.
- void addLongVarchar ()

  Adds a column of type LONGVARCHAR.
- void addNumeric ()

  Adds a column of type NUMERIC.

- void addTime ()

  Adds a column of type TIME.
- void addTimestamp ()

  Adds a column of type TIMESTAMP.
- void addTimestampTz ()

  Adds a column of type TIMESTAMP WITH TIMEZONE.
- void addTimeTz ()

  Adds a column of type TIME WITH TIMEZONE.
- void addVarbinary ()

  Adds a column of type VARBINARY.
- void addVarchar ()

  Adds a column of type VARCHAR.

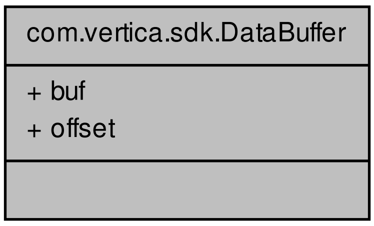
Detailed Description

Represents (unsized) types of the columns used as input/output of a User Defined Function/Transform Function.

This class is used only for generating the function or transform function prototype, where the sizes and/or precisions of the data types are not known.

com.vertica.sdk.DataBuffer Class Reference

Collaboration diagram for com.vertica.sdk.DataBuffer:



Public Attributes

- byte[] buf

  buffer
- int offset

  Size of the buffer in bytes.

Detailed Description

DataBuffer

A contiguous in-memory buffer

Member Data Documentation

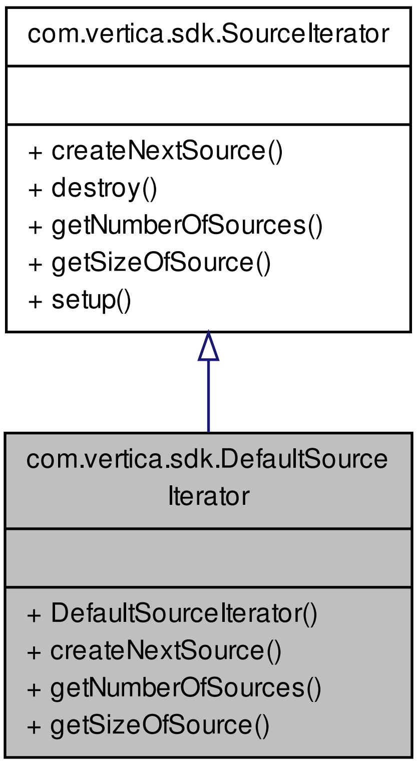
int com.vertica.sdk.DataBuffer.offset

Size of the buffer in bytes.

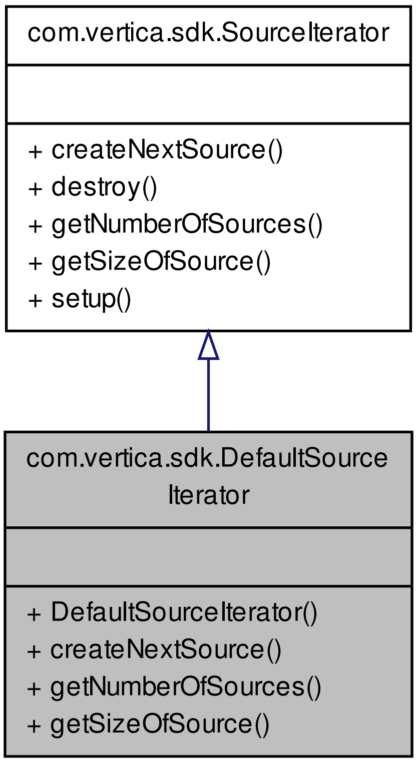
Number of bytes that have been processed by the UDL

com.vertica.sdk.DefaultSourceIterator Class Reference

Inheritance diagram for com.vertica.sdk.DefaultSourceIterator:



Collaboration diagram for com.vertica.sdk.DefaultSourceIterator:



Public Member Functions

- **DefaultSourceIterator** (ArrayList< UDSource > sources)
- UnsizedUDSource createNextSource (ServerInterface srvInterface)

  Create the next UDSource to process.
- void destroy (ServerInterface srvInterface, NodeSpecifyingPlanContext planCtxt) throws UdfException

  Tear down this SourceIterator.
- int getNumberOfSources ()
- Integer **getSizeOfSource** (int sourceNum)
- void setup (ServerInterface srvInterface, NodeSpecifyingPlanContext planCtxt) throws UdfException

  Set up this SourceIterator.

Member Function Documentation

UnsizedUDSource com.vertica.sdk.DefaultSourceIterator.createNextSource (ServerInterface *srvInterface*) `[virtual]`

Create the next UDSource to process.

Should return NULL if no further sources are available for processing.

Note that the previous Source may still be open and in use on a different thread when this function is called.

Returns

a new Source instance corresponding to a new input stream

Exceptions

UdfException	

Implements com.vertica.sdk.SourceIterator.

void com.vertica.sdk.SourceIterator.destroy (ServerInterface *srvInterface,* NodeSpecifyingPlanContext *planCtxt*) throws UdfException `[inherited]`

Tear down this SourceIterator.

Should perform clean-up

Exceptions

UdfException	

int com.vertica.sdk.DefaultSourceIterator.getNumberOfSources ( ) `[virtual]`

Returns

the total number of Sources that this factory will produce

Exceptions

UdfException	

Implements com.vertica.sdk.SourceIterator.

void com.vertica.sdk.SourceIterator.setup (ServerInterface *srvInterface,* NodeSpecifyingPlanContext *planCtxt*) throws UdfException `[inherited]`

Set up this SourceIterator.

Should perform setup that should not take place in the constructor due to the exception-handling semantics of constructors

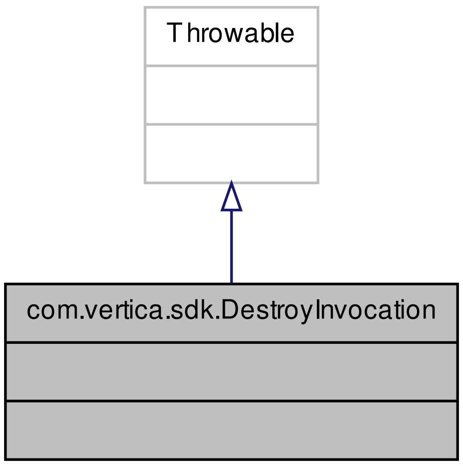
Exceptions

UdfException	

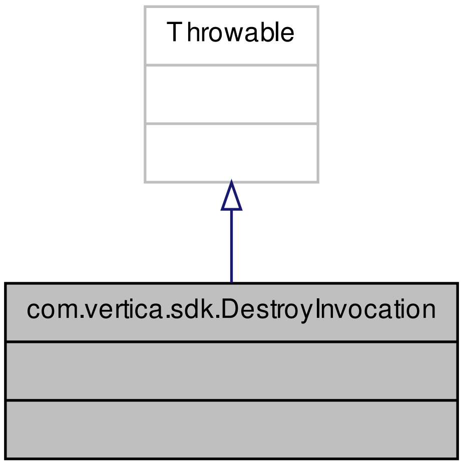
com.vertica.sdk.DestroyInvocation Class Reference

Used to support canceling UDx and invoking the UDx's destroy call back function. This exception is thrown when Vertica needs to cancel the running UDx to jump out of current control flow.

Inheritance diagram for com.vertica.sdk.DestroyInvocation:



Collaboration diagram for com.vertica.sdk.DestroyInvocation:



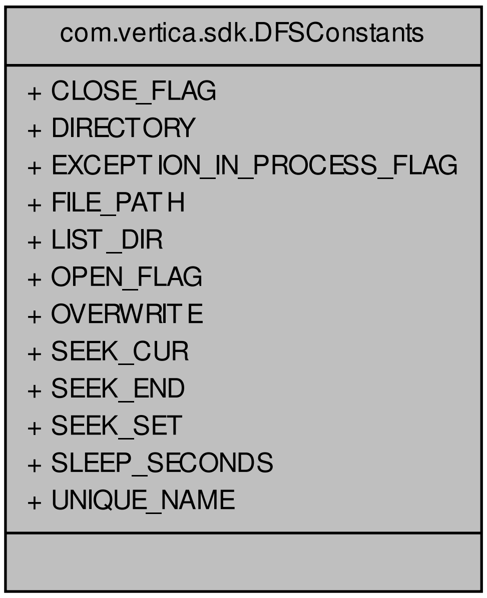
Detailed Description

Used to support canceling UDx and invoking the UDx's destroy call back function. This exception is thrown when Vertica needs to cancel the running UDx to jump out of current control flow.

For UDx to work properly, please do not catch and handle this exception

com.vertica.sdk.DFSConstants Class Reference

Collaboration diagram for com.vertica.sdk.DFSConstants:

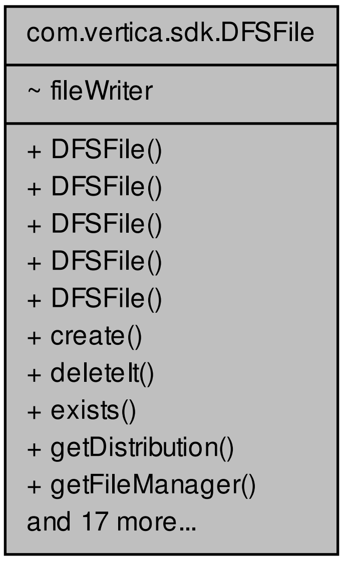


Static Public Attributes

- static final String **CLOSE_FLAG** = "isCloseInProcess"
- static final String **DIRECTORY** = "is_directory"
- static final String **EXCEPTION_IN_PROCESS_FLAG** = "isThrowErrorInProcess"
- static final String **FILE_PATH** = "file_path"
- static final String **LIST_DIR** = "is_list_dir"
- static final String **OPEN_FLAG** = "isBeforeProcess"
- static final String **OVERWRITE** = "is_overwrite"
- static final int **SEEK_CUR** = 1
- static final int **SEEK_END** = 2
- static final int **SEEK_SET** = 0
- static final String **SLEEP_SECONDS** = "sleep_seconds"
- static final String **UNIQUE_NAME** = "is_unique_name"

com.vertica.sdk.DFSFile Class Reference

Collaboration diagram for com.vertica.sdk.DFSFile:



Classes

- enum DFSDistribution
- enum DFSScope

Public Member Functions

- DFSFile ()
- **DFSFile** (ServerInterface srvInterface)
- **DFSFile** (ServerInterface srvInterface, String fName) throws UdfException, DestroyInvocation
- **DFSFile** (String fName, FileManager fmgr) throws UdfException, DestroyInvocation
- **DFSFile** (String fName, FileManager fmgr, boolean is_dir, boolean is_file, boolean exists, long fSize) throws UdfException, DestroyInvocation
- void **create** (DFSScope dfsScope, DFSDistribution dfsDistrib) throws UdfException, DestroyInvocation
- int deleteIt (boolean isRecursively) throws UdfException, DestroyInvocation
- boolean **exists** () throws UdfException
- DFSDistribution **getDistribution** ()
- FileManager **getFileManager** ()
- long **getFileWriter** ()
- String **getName** ()
- DFSScope **getScope** ()
- ServerInterface **getServerInterface** ()
- long **getSize** ()
- DFSFileStatus **getStatus** ()
- boolean **isDir** ()

- boolean **isFile** ()
- List< DFSFile > listFiles () throws UdfException, DestroyInvocation
- void **setDir** (boolean thisIsaDirectory)
- void **setDistribution** (DFSDistribution dfsDist)
- void **setFile** (boolean thisIsaFile)
- void **setFileManager** (ServerInterface srvInterface)
- void setName (String fName) throws UdfException, DestroyInvocation
- void **setScope** (DFSScope dfsScope)
- void **setSize** (long fSize)
- void **setStatus** (DFSFileStatus dfsStatus)

Detailed Description

The main class used by users to initiate DFS operations

Constructor & Destructor Documentation

com.vertica.sdk.DFSFile.DFSFile ( )

DFSFile INITIATION IS ONLY AVAILABLE DURING THE PLANNING/SETUP AND FINALIZE/DESTROY PHASES OF A PLAN. NOT AVAILABLE DURING EXECUTION/PROCESSING.

Member Function Documentation

int com.vertica.sdk.DFSFile.deleteIt (boolean *isRecursively*) throws UdfException, DestroyInvocation

Deletes a DFS file.

Returns

0 is successful, throw exceptions if there are errors.

List<DFSFile> com.vertica.sdk.DFSFile.listFiles ( ) throws UdfException, DestroyInvocation

Lists files under the path specified by 'fileName'

Returns

a list of DFSFile found under the path.

void com.vertica.sdk.DFSFile.setName (String *fName*) throws UdfException, DestroyInvocation

Renames file identified by 'srcFilePath' to 'destFilePath' returns 0, throws exceptions if there are errors public int rename(String newName) throws UdfException { validateFileOrThrow(); return fileManager.rename(fileName, newName); }

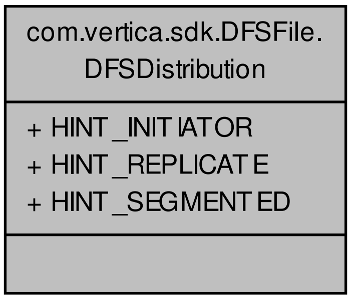
Copy a file/directory from 'srcFilePath' to 'destFilePath'. returns 0, throws exceptions if there are errors. public int copy(DFSFile dfsFile, boolean isRecursively) throws UdfException { validateFileOrThrow(); return fileManager.copy(fileName, dfsFile.getName(), isRecursively);

}

Make a directory, identified by 'dirPath' returns 0, throws exceptions if there are errors. public int makeDir() throws UdfException { validateFileOrThrow(); return fileManager.makeDir(fileName); }

com.vertica.sdk.DFSFile.DFSDistribution Enum Reference

Collaboration diagram for com.vertica.sdk.DFSFile.DFSDistribution:



Public Attributes

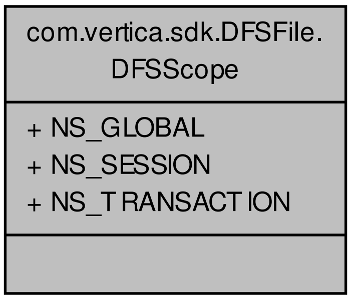
- **HINT_INITIATOR**
- **HINT_REPLICATE**
- **HINT_SEGMENTED**

Detailed Description

Defines how a file is replicated across nodes in the cluster. Used at the file creation time.

com.vertica.sdk.DFSFile.DFSScope Enum Reference

Collaboration diagram for com.vertica.sdk.DFSFile.DFSScope:



Public Attributes

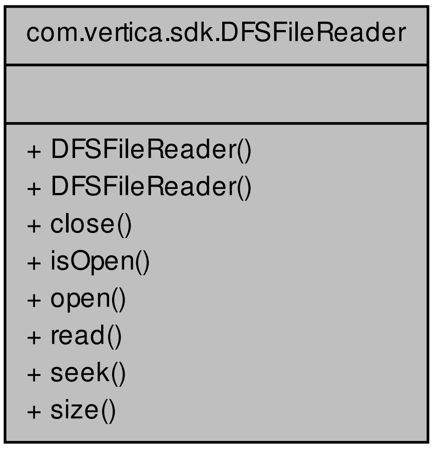
- **NS_GLOBAL**
- **NS_SESSION**
- **NS_TRANSACTION**

Detailed Description

Defines the scope fo the file. Used at the file creation time.

com.vertica.sdk.DFSFileReader Class Reference

Collaboration diagram for com.vertica.sdk.DFSFileReader:



Public Member Functions

- **DFSFileReader** (DFSFile dfsFile)
- void close () throws UdfException, DestroyInvocation
- boolean **isOpen** ()
- void **open** () throws UdfException, DestroyInvocation
- int read (ByteBuffer buffer, int size) throws UdfException, DestroyInvocation
- long seek (long offset, int origin) throws UdfException, DestroyInvocation
- long **size** ()

Member Function Documentation

void com.vertica.sdk.DFSFileReader.close ( ) throws **UdfException, DestroyInvocation**

Closes the file opened for reading

int com.vertica.sdk.DFSFileReader.read (ByteBuffer *buffer*, int *size*) throws UdfException, DestroyInvocation

Reads 'size' of bytes into buffer pointed by 'ptr' from the file opened for reading.

Returns

number of bytes read, 0 if no bytes were read, indicates the EOF. throws exceptions if there are errors.

long com.vertica.sdk.DFSFileReader.seek (long *offset*, int *origin*) throws UdfException, DestroyInvocation

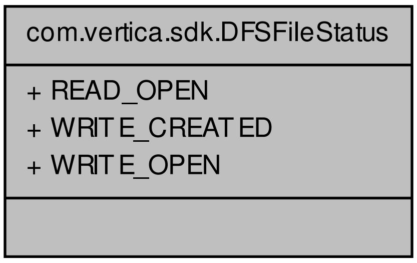
Reposition the read file offset.

Returns

the new file offset.

com.vertica.sdk.DFSFileStatus Enum Reference

Collaboration diagram for com.vertica.sdk.DFSFileStatus:



Public Attributes

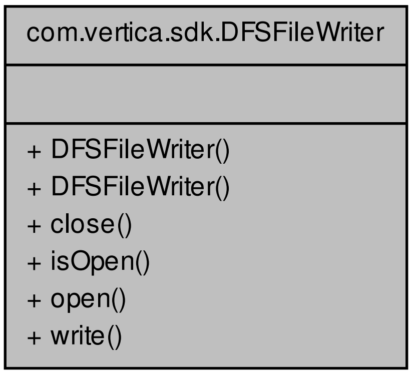
- **READ_OPEN**
- **WRITE_CREATED**
- **WRITE_OPEN**

Detailed Description

Internal DFSFile status to indicate its state

com.vertica.sdk.DFSFileWriter Class Reference

Collaboration diagram for com.vertica.sdk.DFSFileWriter:



Public Member Functions

- **DFSFileWriter** (DFSFile dfsFile)
- void close () throws UdfException, DestroyInvocation
- boolean **isOpen** ()
- void open () throws UdfException, DestroyInvocation
- int write (ByteBuffer buffer) throws UdfException, DestroyInvocation

Member Function Documentation

void com.vertica.sdk.DFSFileWriter.close ( ) throws **UdfException, DestroyInvocation**

Closes the file opened for writing.

void com.vertica.sdk.DFSFileWriter.open ( ) throws **UdfException, DestroyInvocation**

Opens a file for writing.

int com.vertica.sdk.DFSFileWriter.write (ByteBuffer *buffer*) throws **UdfException, DestroyInvocation**

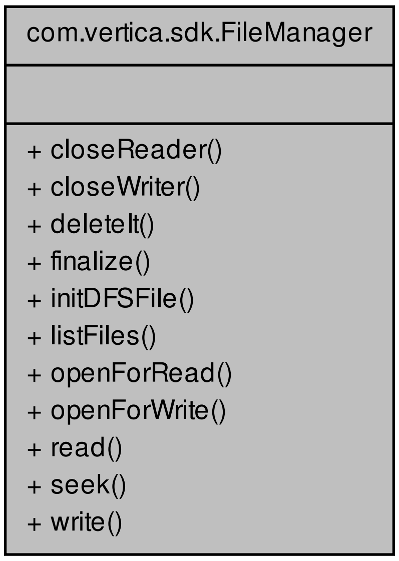
Writes bytes into the file from the ByteBuffer pointed by 'buffer'. Bytes are retrieved from the buffer starting from the current position till it's limit. Current position will be advanced depending on how many bytes are written.

Returns

number of bytes written, could be 0.

com.vertica.sdk.FileManager Class Reference

Collaboration diagram for com.vertica.sdk.FileManager:



Public Member Functions

- abstract void closeReader (long readerID) throws UdfException, DestroyInvocation
- abstract void closeWriter (long writerID) throws UdfException, DestroyInvocation
- abstract int deleteIt (String fileName, boolean isRecursively) throws UdfException, DestroyInvocation
- abstract void finalize ()
- abstract boolean initDFSFile (DFSFile file) throws UdfException, DestroyInvocation
- abstract List< DFSFile > listFiles (String fileName) throws UdfException, DestroyInvocation
- abstract long openForRead (String fileName) throws UdfException, DestroyInvocation
- abstract long openForWrite (String fileName, DFSFile.DFSScope dfsScope, DFSFile.DFSDistribution dfs-Distrib) throws UdfException, DestroyInvocation
- abstract int read (long readerID, ByteBuffer buffer, int size) throws UdfException, DestroyInvocation
- abstract long seek (long readerID, long offset, int origin) throws UdfException, DestroyInvocation
- abstract int write (long writerID, ByteBuffer buffer) throws UdfException, DestroyInvocation

Detailed Description

File Manager is a session level co-ordinator, which will be used by DFSFile, DFSFileReader and DFSFileWriter to interact with Catalog and Storage system of Vertica.

Member Function Documentation

abstract void com.vertica.sdk.FileManager.closeReader (long *readerID*) throws UdfException, DestroyInvocation `[pure virtual]`

Closes the file opened for reading, identified by 'readerID'

abstract void com.vertica.sdk.FileManager.closeWriter (long *writerID*) throws UdfException, DestroyInvocation `[pure virtual]`

Closes teh file opened for writing, identified by 'writerID'

abstract int com.vertica.sdk.FileManager.deleteIt (String *fileName,* boolean *isRecursively*) throws UdfException, DestroyInvocation `[pure virtual]`

Deletes a DFSFile, identified by full path 'fileName'.

Returns

0 if successful, throw exceptions if there are errors

abstract void com.vertica.sdk.FileManager.finalize ( ) `[pure virtual]`

Renames file identified by 'srcFilePath' to 'destFilePath' returns 0, throws exceptions if there are errors. public abstract int rename(String srcFilePath, String destFilePath);

Copy a file/directory from 'srcFilePath' to 'destFilePath'. returns 0, throws exceptions if there are errors. public abstract int copy(String srcFilePath, String destFilePath, boolean isRecursively);

Make a directory, identified by 'dirPath' returns 0, throws exceptions if there are errors public abstract int makeDir(-String dirPath); Finalizes a plan/query/statement. Should only invoke on the initiator node of a query. Complete file replication and commit metadata into the catalog. returns nothing, throws exceptions if there are errors.

abstract boolean com.vertica.sdk.FileManager.initDFSFile (DFSFile *file*) throws UdfException, DestroyInvocation `[pure virtual]`

Initialize a DFSFile upon constructing. returns true if file exists in the DFS, false otherwise, throws exceptions if there are errors.

abstract List<DFSFile> com.vertica.sdk.FileManager.listFiles (String *fileName*) throws UdfException, DestroyInvocation `[pure virtual]`

Lists file under the path specified by 'fileName'

Returns

a list of DFSFile found under the path.

abstract long com.vertica.sdk.FileManager.openForRead (String *fileName*) throws UdfException, DestroyInvocation `[pure virtual]`

Opens a file for reading

Returns

A unique identifier for the file opened. Return value is less than 0 if there are errors

abstract long com.vertica.sdk.FileManager.openForWrite (String *fileName,* DFSFile.DFSScope *dfsScope,* DFSFile.DFSDistribution *dfsDistrib*) throws UdfException, DestroyInvocation `[pure virtual]`

Opens a file for writing

Returns

A unique identifier for the file opened. Return value is less tan 0 if there are errors

abstract int com.vertica.sdk.FileManager.read (long *readerID,* ByteBuffer *buffer,* int *size*) throws UdfException, DestroyInvocation `[pure virtual]`

Reads 'size' of bytes into buffer from the file identified by 'readerID'.

Returns

number of bytes read, 0 if no bytes were read, indicates the EOF. throws exceptions if there are errors

abstract long com.vertica.sdk.FileManager.seek (long *readerID,* long *offset,* int *origin*) throws UdfException, DestroyInvocation `[pure virtual]`

Reposition the read file offset

Returns

the new file offset.

abstract int com.vertica.sdk.FileManager.write (long *writerID,* ByteBuffer *buffer*) throws UdfException, DestroyInvocation `[pure virtual]`

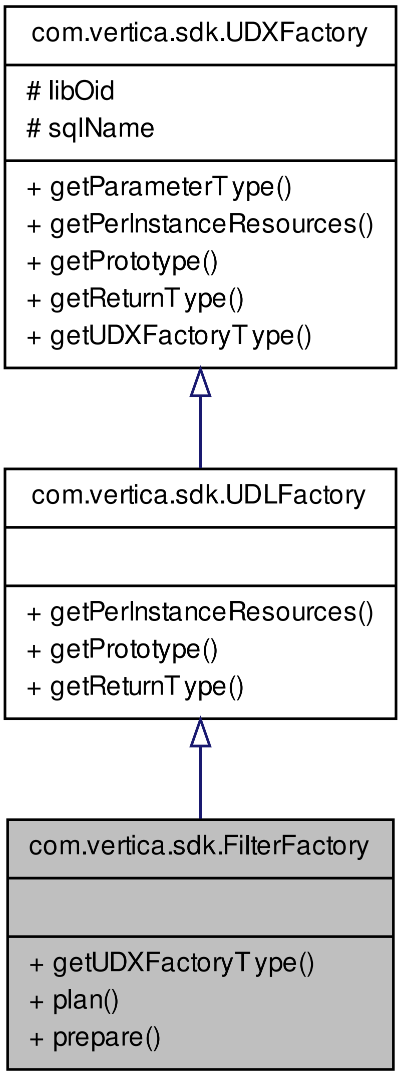
Writes bytes into the file identified by 'writerID' from the buffer. Bytes are retrieved from buffer starting from the current position till it's limit. Current position will be advanced depending on how many bytes are written.

Returns

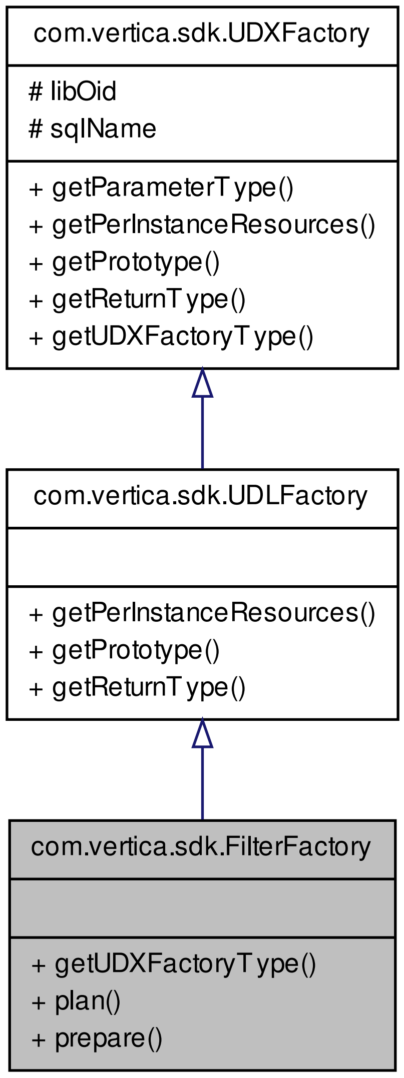
number of bytes written, less than 0 if there are any errors.

com.vertica.sdk.FilterFactory Class Reference

Inheritance diagram for com.vertica.sdk.FilterFactory:



Collaboration diagram for com.vertica.sdk.FilterFactory:



Public Member Functions

- void getParameterType (ServerInterface srvInterface, SizedColumnTypes parameterTypes)
- void **getPerInstanceResources** (ServerInterface srvInterface, VResources res)
- void getPrototype (ServerInterface srvInterface, ColumnTypes argTypes, ColumnTypes returnType)
- void getReturnType (ServerInterface srvInterface, SizedColumnTypes argTypes, SizedColumnTypes return-Type)
- UDXType getUDXFactoryType ()
- void plan (ServerInterface srvInterface, PlanContext planCtxt) throws UdfException
- abstract UDFilter prepare (ServerInterface srvInterface, PlanContext planCtxt) throws UdfException

Protected Attributes

- long **libOid**
- String **sqlName**

Detailed Description

Construct a single Filter.

Note that FilterFactories are singletons. Subclasses should be stateless, with no fields containing data, just methods. plan() and prepare() methods must never modify any global variables or state; they may only modify the variables that they are given as arguments.

Member Function Documentation

void com.vertica.sdk.UDXFactory.getParameterType (ServerInterface *srvInterface,* SizedColumnTypes *parameterTypes*) `[inherited]`

Function to tell Vertica the name and types of parameters that this function uses. Vertica will use this to warn function callers that certain parameters they provide are not affecting anything, or that certain parameters that are not being set are reverting to default values.

void com.vertica.sdk.UDLFactory.getPrototype (ServerInterface *srvInterface,* ColumnTypes *argTypes,* ColumnTypes *returnType*) `[virtual]`, `[inherited]`

Provides the argument and return types of the UDL. UDL's take no input tuples; as such, their prototype is empty.

Implements com.vertica.sdk.UDXFactory.

void com.vertica.sdk.UDLFactory.getReturnType (ServerInterface *srvInterface,* SizedColumnTypes *argTypes,* SizedColumnTypes *returnType*) `[virtual]`, `[inherited]`

Not used in this form

Implements com.vertica.sdk.UDXFactory.

UDXType com.vertica.sdk.FilterFactory.getUDXFactoryType ( ) `[virtual]`

Returns

the type of UDX Object instance this factory returns.

Note

User subclasses should use the appropriate subclass of UDXFactory and not override this method on their own.

Implements com.vertica.sdk.UDXFactory.

void com.vertica.sdk.FilterFactory.plan (ServerInterface *srvInterface,* PlanContext *planCtxt*) throws UdfException

Execute any planning logic required at query plan time. This method is run once per query, during query initialization. Its job is to perform parameter validation, and to modify the set of nodes that the COPY statement will run on (through srvInterface).

plan() runs exactly once per query, on the initiator node. If it throws an exception, the query will not proceed; it will be aborted prior to distributing the query to the other nodes and running prepare().

Parameters

srvInterface	Interface to server operations and functionality, including (not-per-column) parameter lookup
planCtxt	Context for storing and retrieving arbitrary data, for use just by this instance of this query. The same context is shared with plan().

Exceptions

UdfException	

abstract UDFilter com.vertica.sdk.FilterFactory.prepare (ServerInterface *srvInterface,* PlanContext *planCtxt*) throws UdfException `[pure virtual]`

Initialize a UDFilter. This function will be called on each node, prior to the Load operator starting to execute.

Parameters

srvInterface	Interface to server operations and functionality, including (not-per-column) parameter lookup
planCtxt	Context for storing and retrieving arbitrary data, for use just by this instance of this query. The same context is shared with plan().

Returns

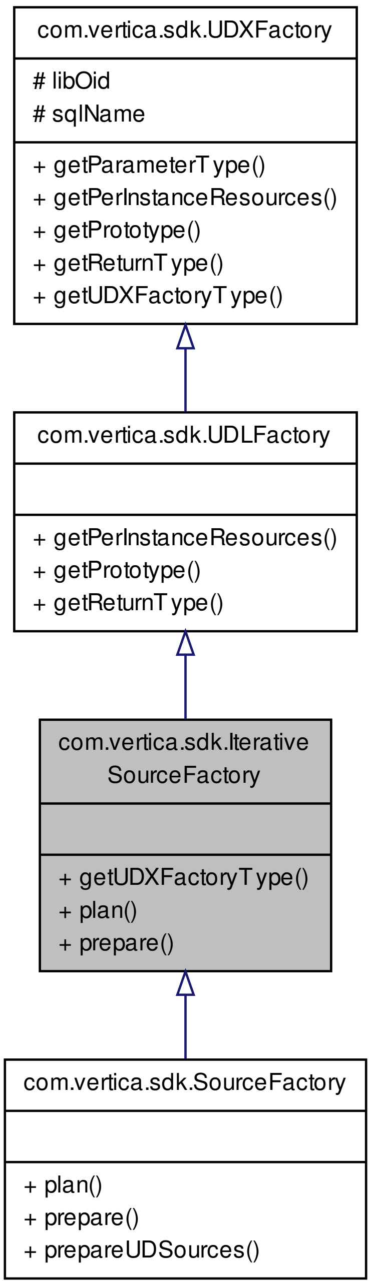
UDFilter instance to use for this query

Exceptions

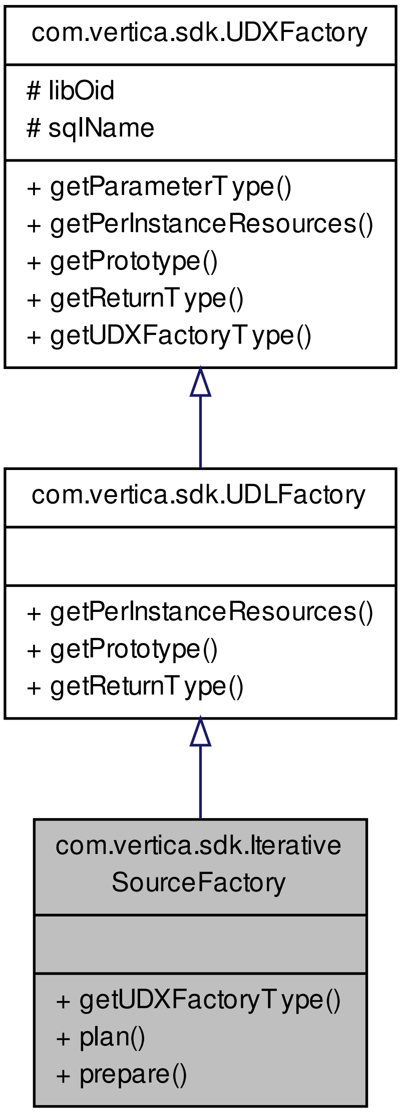
UdfException	

com.vertica.sdk.IterativeSourceFactory Class Reference

Inheritance diagram for com.vertica.sdk.IterativeSourceFactory:



Collaboration diagram for com.vertica.sdk.IterativeSourceFactory:



Public Member Functions

- void getParameterType (ServerInterface srvInterface, SizedColumnTypes parameterTypes)
- void **getPerInstanceResources** (ServerInterface srvInterface, VResources res)
- void getPrototype (ServerInterface srvInterface, ColumnTypes argTypes, ColumnTypes returnType)
- void getReturnType (ServerInterface srvInterface, SizedColumnTypes argTypes, SizedColumnTypes return-Type)
- UDXType getUDXFactoryType ()
- void plan (ServerInterface srvInterface, NodeSpecifyingPlanContext planCtxt) throws UdfException
- abstract SourceIterator prepare (ServerInterface srvInterface, NodeSpecifyingPlanContext planCtxt) throws UdfException

Protected Attributes

- long **libOid**
- String **sqlName**

Detailed Description

High-level initialization required by a UDSource.

Performs initial validation and planning of the query, before it is distributed over the network. Also instantiates objects to perform further initialization on each node, once the query has been distributed.

Note that SourceFactories are singletons. Subclasses should be stateless, with no fields containing data, just methods. plan() and prepare() methods must never modify any global variables or state; they may only modify the variables that they are given as arguments. (If global state must be modified, use SourceIterator.)

Member Function Documentation

void com.vertica.sdk.UDXFactory.getParameterType (ServerInterface *srvInterface*, SizedColumnTypes *parameterTypes*) `[inherited]`

Function to tell Vertica the name and types of parameters that this function uses. Vertica will use this to warn function callers that certain parameters they provide are not affecting anything, or that certain parameters that are not being set are reverting to default values.

void com.vertica.sdk.UDLFactory.getPrototype (ServerInterface *srvInterface*, ColumnTypes *argTypes*, ColumnTypes *returnType*) `[virtual]`, `[inherited]`

Provides the argument and return types of the UDL. UDL's take no input tuples; as such, their prototype is empty.

Implements com.vertica.sdk.UDXFactory.

void com.vertica.sdk.UDLFactory.getReturnType (ServerInterface *srvInterface*, SizedColumnTypes *argTypes*, SizedColumnTypes *returnType*) `[virtual]`, `[inherited]`

Not used in this form

Implements com.vertica.sdk.UDXFactory.

UDXType com.vertica.sdk.IterativeSourceFactory.getUDXFactoryType ( ) `[virtual]`

Returns

the type of UDX Object instance this factory returns.

Note

User subclasses should use the appropriate subclass of UDXFactory and not override this method on their own.

Implements com.vertica.sdk.UDXFactory.

void com.vertica.sdk.IterativeSourceFactory.plan (ServerInterface *srvInterface,* NodeSpecifyingPlanContext *planCtxt*) throws UdfException

Execute any planning logic required at query plan time. This method is run once per query, during query initialization. Its job is to perform parameter validation, and to modify the set of nodes that the COPY statement will run on.

plan() runs exactly once per query, on the initiator node. If it throws an exception, the query will not proceed; it will be aborted prior to distributing the query to the other nodes and running prepare().

Exceptions

UdfException	

abstract SourceIterator com.vertica.sdk.IterativeSourceFactory.prepare (ServerInterface *srvInterface,* NodeSpecifyingPlanContext *planCtxt*) throws UdfException `[pure virtual]`

Prepare this SourceFactory to start creating sources. This function will be called on each node, prior to the Load operator starting to execute and prior to any other virtual functions on this class being called.

If necessary, it is safe for this method to store any of the argument references as local fields on this instance. All will persist for the duration of the query.

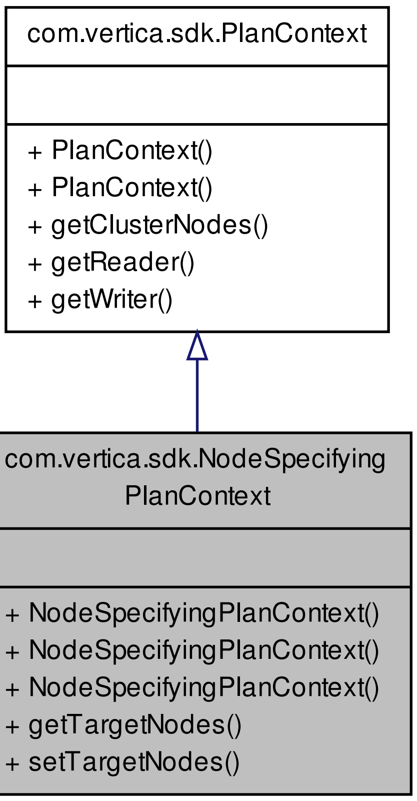
Exceptions

UdfException	

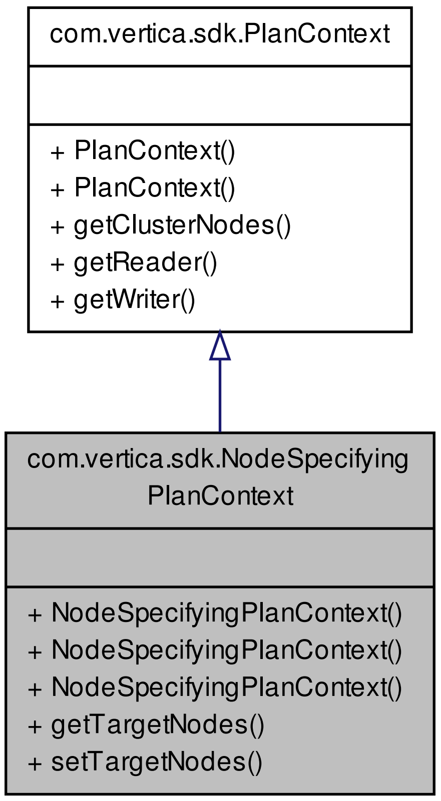
Implemented in com.vertica.sdk.SourceFactory.

com.vertica.sdk.NodeSpecifyingPlanContext Class Reference

Inheritance diagram for com.vertica.sdk.NodeSpecifyingPlanContext:


com.vertica.sdk.PlanContext
+ PlanContext()
+ PlanContext()
+ getClusterNodes()
+ getReader()
+ getWriter()
com.vertica.sdk.NodeSpecifying PlanContext
+ NodeSpecifyingPlanContext()
+ NodeSpecifyingPlanContext()
+ NodeSpecifyingPlanContext()
+ getTargetNodes()
+ setTargetNodes()

Collaboration diagram for com.vertica.sdk.NodeSpecifyingPlanContext:


com.vertica.sdk.PlanContext
+ PlanContext()
+ PlanContext()
+ getClusterNodes()
+ getReader()
+ getWriter()
com.vertica.sdk.NodeSpecifying PlanContext
+ NodeSpecifyingPlanContext()
+ NodeSpecifyingPlanContext()
+ NodeSpecifyingPlanContext()
+ getTargetNodes()
+ setTargetNodes()

Public Member Functions

- **NodeSpecifyingPlanContext** (ParamWriter writer, ArrayList< String > clusterNodes, ArrayList< String > targetNodes)
- **NodeSpecifyingPlanContext** (ParamWriter writer, ArrayList< String > clusterNodes)
- **NodeSpecifyingPlanContext** (ParamWriter writer)
- ArrayList< String > getClusterNodes ()
- ParamReader getReader ()
- ArrayList< String > getTargetNodes ()
- ParamWriter getWriter ()
- void setTargetNodes (ArrayList< String > nodes) throws UdfException

Detailed Description

Interface that allows storage of query-plan state, when different parts of query planning take place on different computers. For example, if some work is done on the query initiator node and some is done on each node executing the query.

In addition to the functionality provided by PlanContext, NodeSpecifyingPlanContext allows you to specify which nodes the query should run on.

Member Function Documentation

ArrayList<String> com.vertica.sdk.PlanContext.getClusterNodes ( ) `[inherited]`

Get a list of all of the nodes in the current cluster, by node name

ParamReader com.vertica.sdk.PlanContext.getReader ( ) `[inherited]`

Get a read-only instance of the current context

ArrayList<String> com.vertica.sdk.NodeSpecifyingPlanContext.getTargetNodes ( )

Return the set of nodes that this query is currently set to run on

ParamWriter com.vertica.sdk.PlanContext.getWriter ( ) `[inherited]`

Get the current context for writing

void com.vertica.sdk.NodeSpecifyingPlanContext.setTargetNodes (ArrayList< String > ***nodes*** **) throws UdfException**

Change the set of nodes that the query is intended to run on. Throws if any of the specified node names is not actually the name of any node in the cluster.

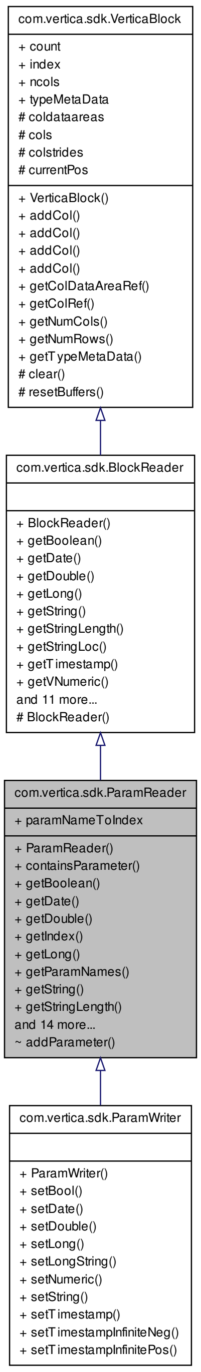
Exceptions

UdfException	

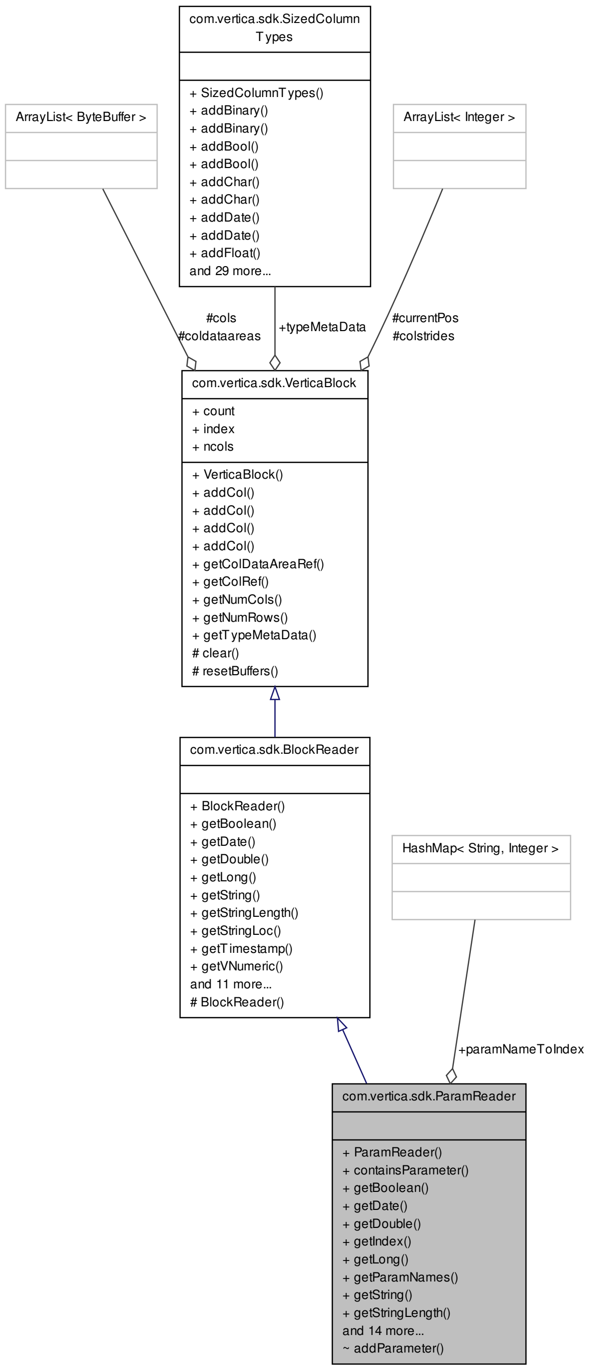
com.vertica.sdk.ParamReader Class Reference

A wrapper around Parameters that have a name->value correspondence.

Inheritance diagram for com.vertica.sdk.ParamReader:



Collaboration diagram for com.vertica.sdk.ParamReader:



Public Member Functions

- void addCol (ByteBuffer arg, int colstride, VerticaType dt, String colName)
- void addCol (ByteBuffer arg, int colstride, VerticaType dt)
- void addCol (ByteBuffer arg, ByteBuffer da, int colstride, VerticaType dt)
- void addCol (ByteBuffer arg, ByteBuffer da, int colstride, VerticaType dt, String colName)
- boolean containsParameter (String paramName)

Function to see if the ParamReader has a value for the parameter.

- boolean getBoolean (int idx)

  Get a BOOLEAN value from the input row.
- boolean **getBoolean** (String paramName) throws UdfException
- ByteBuffer getColDataAreaRef (int idx)
- ByteBuffer getColRef (int idx)
- java.sql.Date **getDate** (String paramName) throws UdfException
- java.sql.Date getDate (int idx)

  Get a DATE value from the input row.
- double getDouble (int idx)

  Get a DOUBLE value from the input row.
- double **getDouble** (String paramName) throws UdfException
- int **getIndex** (String paramName) throws UdfException
- long getLong (int idx)

  Get a LONG INTEGER value from the input row.
- long **getLong** (String paramName) throws UdfException
- int getNumCols ()
- int getNumRows ()
- ArrayList$<$ String $>$ getParamNames ()

  Return all names of parameters stored in this ParamReader.
- String **getString** (String paramName) throws UdfException
- String getString (int idx)

  Get a reference to an VARCHAR/CHAR/VARBINARY/BINARY value from the input row.
- int **getStringLength** (String paramName) throws UdfException
- int getStringLength (int idx)

  Get length of the String from the input row.
- int getStringLoc (int idx)

  Get 'location' of the String from the input row.
- java.sql.Timestamp **getTimestamp** (String paramName) throws UdfException
- java.sql.Timestamp getTimestamp (int idx)

  Get a TIMESTAMP value from the input row.
- VerticaType getType (String paramName) throws UdfException

  Return the type of the given parameter.
- SizedColumnTypes getTypeMetaData ()
- VNumeric **getVNumeric** (String paramName) throws UdfException
- VNumeric getVNumeric (int idx)

  Get a reference to a VNumeric value from the input row.
- VString **getVString** (String paramName) throws UdfException
- VString getVString (int idx)

  Get a reference from the input row to an VString value, which represents a SQL VARCHAR/CHAR/VARBINARY/BINARY value.
- boolean isBooleanNull (int idx)

  Check whether a value from the input row is NULL in BOOLEAN type.
- boolean **isBooleanNull** (String paramName) throws UdfException
- boolean **isDateNull** (String paramName) throws UdfException
- boolean isDateNull (int idx)

  Check whether a value from the input row is NULL in DATE type.
- boolean isDoubleNull (int idx)

  Check whether a value from the input row is NULL in DOUBLE type.
- boolean **isDoubleNull** (String paramName) throws UdfException
- boolean isEmpty ()

  Returns true if there are no parameters.

- boolean isLongNull (int idx)

  Check whether a value from the input row is NULL in LONG INTERGER type.
- boolean **isLongNull** (String paramName) throws UdfException
- boolean **isStringNull** (String paramName) throws UdfException
- boolean isStringNull (int idx)

  Check whether a value from the input row is NULL in SQL VARCHAR/CHAR/VARBINARY/BINARY type.
- boolean **isTimestampInfinite** (String paramName) throws UdfException
- boolean isTimestampInfinite (int idx)

  Check whether a TIMESTAMP value from the input row represents 'infinity'.
- boolean **isTimestampInfiniteNeg** (String paramName) throws UdfException
- boolean isTimestampInfiniteNeg (int idx)

  Check whether a TIMESTAMP value from the input row represents '-infinity'.
- boolean **isTimestampInfinitePos** (String paramName) throws UdfException
- boolean isTimestampInfinitePos (int idx)

  Check whether a TIMESTAMP value from the input row represents '+infinity'.
- boolean **isTimestampNull** (String paramName) throws UdfException
- boolean isTimestampNull (int idx)

  Check whether a value from the input row is NULL in TIMESTAMP type.
- boolean next () throws UdfException, DestroyInvocation

Public Attributes

- int **count**
- int **index**
- int **ncols**
- HashMap< String, Integer > **paramNameToIndex**
- SizedColumnTypes **typeMetaData**

Protected Member Functions

- void **clear** ()
- void **resetBuffers** ()

Protected Attributes

- ArrayList< ByteBuffer > **coldataareas**
- ArrayList< ByteBuffer > **cols**
- ArrayList< Integer > **colstrides**
- ArrayList< Integer > **currentPos**

Detailed Description

A wrapper around Parameters that have a name->value correspondence.

Member Function Documentation

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, int *colstride*, VerticaType *dt*, String *colName*) `[inherited]`

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
colstride	The stride between data instances.
dt	The type of input.
colName	Name of the column

Referenced by com.vertica.sdk.VerticaBlock.addCol().

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, int *colstride*, VerticaType *dt*) `[inherited]`

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
colstride	The stride between data instances.
dt	The type of input.

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, ByteBuffer *da*, int *colstride*, VerticaType *dt*) `[inherited]`

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
da	The location to find out of band string data.
colstride	The stride between data instances.
dt	The type of input.

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, ByteBuffer *da*, int *colstride*, VerticaType *dt*, String *colName*) `[inherited]`

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
da	The location to find out of band string data.
colstride	The stride between data instances.
dt	The type of input.
colName	Name of the column

boolean com.vertica.sdk.BlockReader.getBoolean (int *idx*) `[inherited]`

Get a BOOLEAN value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The value of the idx'th argument, cast as a BOOLEAN.

ByteBuffer com.vertica.sdk.VerticaBlock.getColDataAreaRef (int *idx*) `[inherited]`

Get the ByteBuffer that stores out of line string data (Data Area) for the idx'th argument

Parameters

idx	

Returns

Referenced by com.vertica.sdk.BlockReader.getVString().

ByteBuffer com.vertica.sdk.VerticaBlock.getColRef (int *idx*) `[inherited]`

Returns

a ByteBuffer to the idx'th argument, containing data for the column

Example:

```
* ByteBuffer a = arg_reader.getColPtr(0);
*
```

Referenced by com.vertica.sdk.PartitionWriter.copyFromInput(), com.vertica.sdk.BlockReader.getBoolean(), com.-vertica.sdk.BlockReader.getDouble(), com.vertica.sdk.BlockReader.getLong(), com.vertica.sdk.BlockReader.-getStringLength(), com.vertica.sdk.BlockReader.getStringLoc(), com.vertica.sdk.BlockReader.getVNumeric(), com.vertica.sdk.BlockReader.getVString(), com.vertica.sdk.BlockWriter.getVStringWriter(), com.vertica.sdk.-BlockReader.isBooleanNull(), com.vertica.sdk.ParamWriter.setBool(), com.vertica.sdk.BlockWriter.setBoolean(), com.vertica.sdk.BlockWriter.setBooleanNull(), com.vertica.sdk.ParamWriter.setDouble(), com.vertica.sdk.Block-Writer.setDouble(), com.vertica.sdk.BlockWriter.setDoubleNull(), com.vertica.sdk.PartitionWriter.setLong(), com.-vertica.sdk.BlockWriter.setLongNull(), com.vertica.sdk.ParamWriter.setLongString(), com.vertica.sdk.BlockWriter.-setNumeric(), com.vertica.sdk.ParamWriter.setNumeric(), com.vertica.sdk.BlockWriter.setString(), com.vertica.-sdk.ParamWriter.setString(), and com.vertica.sdk.BlockWriter.setStringNull().

java.sql.Date com.vertica.sdk.BlockReader.getDate (int *idx*) `[inherited]`

Get a DATE value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The value of the idx'th argument, cast as a DATE; null if the column is NULL.

double com.vertica.sdk.BlockReader.getDouble (int *idx*) `[inherited]`

Get a DOUBLE value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The value of the idx'th argument, cast as a DOUBLE.

long com.vertica.sdk.BlockReader.getLong (int *idx*) `[inherited]`

Get a LONG INTEGER value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The value of the idx'th argument, cast as a LONG INTEGER.

Example:

```
* long a = arg_reader.getLong(0);
*
```

Referenced by com.vertica.sdk.BlockReader.getDate(), com.vertica.sdk.BlockReader.getTimestamp(), com.vertica.sdk.BlockReader.isDoubleNull(), com.vertica.sdk.BlockReader.isLongNull(), com.vertica.sdk.BlockReader.isTimestampInfiniteNeg(), and com.vertica.sdk.BlockReader.isTimestampInfinitePos().

int com.vertica.sdk.VerticaBlock.getNumCols ( ) `[inherited]`

Returns

the number of arguments held by this reader.

int com.vertica.sdk.VerticaBlock.getNumRows ( ) `[inherited]`

Returns

the number of rows held by this block.

String com.vertica.sdk.BlockReader.getString (int *idx*) `[inherited]`

Get a reference to an VARCHAR/CHAR/VARBINARY/BINARY value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

a reference to the idx'th argument, cast as an String.

int com.vertica.sdk.BlockReader.getStringLength (int *idx*) `[inherited]`

Get length of the String from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The length of the String in specified column.

Referenced by com.vertica.sdk.BlockReader.getVString(), and com.vertica.sdk.BlockReader.isStringNull().

int com.vertica.sdk.BlockReader.getStringLoc (int *idx*) `[inherited]`

Get 'location' of the String from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The location of the String in specified column. If zero, data is inlined immediately after the header, otherwise data is at offset loc within the data area.

Referenced by com.vertica.sdk.BlockReader.getVString().

java.sql.Timestamp com.vertica.sdk.BlockReader.getTimestamp (int *idx*) `[inherited]`

Get a TIMESTAMP value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The value of the idx'th argument, cast as a TIMESTAMP; null if the column is NULL or represents 'infinity'.

VerticaType com.vertica.sdk.ParamReader.getType (String *paramName*) throws UdfException

Return the type of the given parameter.

Exceptions

UdfException	

SizedColumnTypes com.vertica.sdk.VerticaBlock.getTypeMetaData ( ) `[inherited]`

Returns

information about the types and numbers of arguments

Referenced by com.vertica.sdk.ParamReader.getType().

VNumeric com.vertica.sdk.BlockReader.getVNumeric (int *idx*) `[inherited]`

Get a reference to a VNumeric value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

A reference to the retrieved value cast as a Numeric.

VString com.vertica.sdk.BlockReader.getVString (int *idx*) `[inherited]`

Get a reference from the input row to an VString value, which represents a SQL VARCHAR/CHAR/VARBINARY/B-INARY value.

Parameters

idx	The column number to retrieve from the input row.

Returns

a reference to the idx'th argument, cast as an VString.

Referenced by com.vertica.sdk.BlockReader.getString().

boolean com.vertica.sdk.BlockReader.isBooleanNull (int *idx*) `[inherited]`

Check whether a value from the input row is NULL in BOOLEAN type.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the value is NULL, false otherwise.

boolean com.vertica.sdk.BlockReader.isDateNull (int *idx*) `[inherited]`

Check whether a value from the input row is NULL in DATE type.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the value is NULL, false otherwise.

Referenced by com.vertica.sdk.BlockReader.getDate().

boolean com.vertica.sdk.BlockReader.isDoubleNull (int *idx*) `[inherited]`

Check whether a value from the input row is NULL in DOUBLE type.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the value is NULL, false otherwise.

boolean com.vertica.sdk.BlockReader.isLongNull (int *idx*) `[inherited]`

Check whether a value from the input row is NULL in LONG INTERGER type.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the value is NULL, false otherwise.

Referenced by com.vertica.sdk.BlockReader.isDateNull(), and com.vertica.sdk.BlockReader.isTimestampNull().

boolean com.vertica.sdk.BlockReader.isStringNull (int *idx*) `[inherited]`

Check whether a value from the input row is NULL in SQL VARCHAR/CHAR/VARBINARY/BINARY type.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the value is NULL, false otherwise.

Referenced by com.vertica.sdk.BlockReader.getString().

boolean com.vertica.sdk.BlockReader.isTimestampInfinite (int *idx*) `[inherited]`

Check whether a TIMESTAMP value from the input row represents 'infinity'.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the TIMESTAMP value is '+infinity' or '-infinity', false otherwise.

Referenced by com.vertica.sdk.BlockReader.getTimestamp().

boolean com.vertica.sdk.BlockReader.isTimestampInfiniteNeg (int *idx*) `[inherited]`

Check whether a TIMESTAMP value from the input row represents '-infinity'.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the TIMESTAMP value is '-infinity', false otherwise.

Referenced by com.vertica.sdk.BlockReader.isTimestampInfinite().

boolean com.vertica.sdk.BlockReader.isTimestampInfinitePos (int *idx*) `[inherited]`

Check whether a TIMESTAMP value from the input row represents '+infinity'.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the TIMESTAMP value is '+infinity', false otherwise.

Referenced by com.vertica.sdk.BlockReader.isTimestampInfinite().

boolean com.vertica.sdk.BlockReader.isTimestampNull (int *idx*) `[inherited]`

Check whether a value from the input row is NULL in TIMESTAMP type.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the value is NULL, false otherwise.

Referenced by com.vertica.sdk.BlockReader.getTimestamp().

boolean com.vertica.sdk.BlockReader.next ( ) throws UdfException, DestroyInvocation `[inherited]`

Advance to the next record.

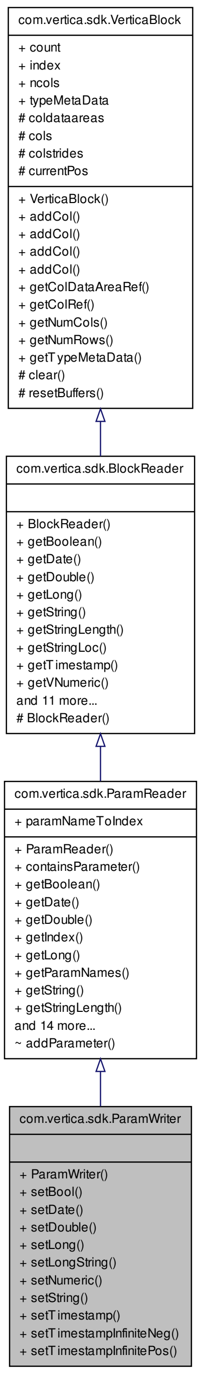
Returns

true if there are more rows to read, false otherwise.

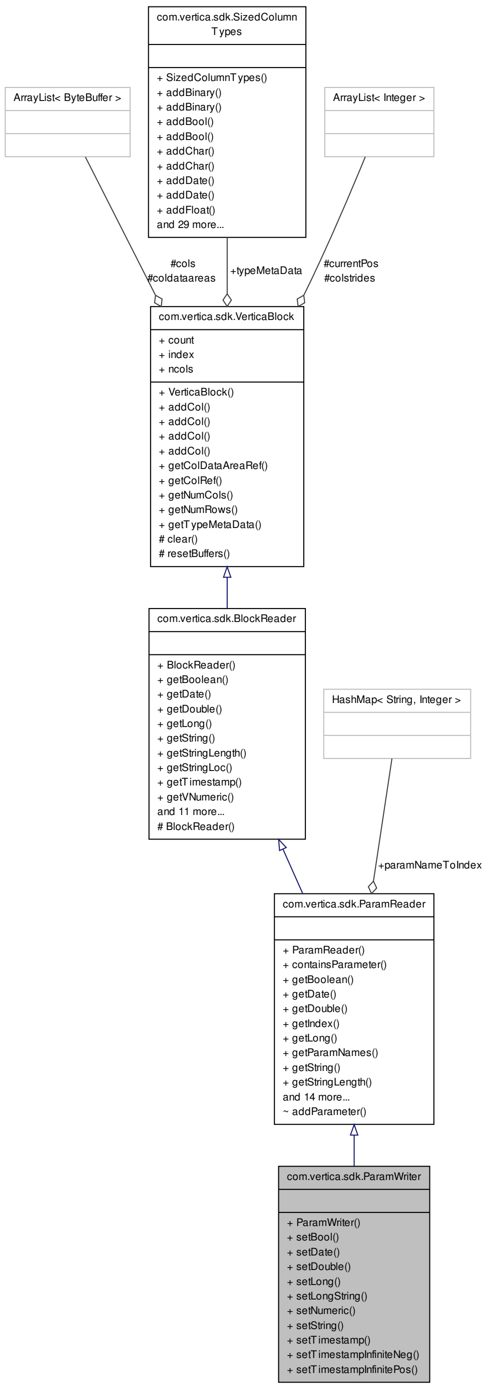
com.vertica.sdk.ParamWriter Class Reference

Iterator interface for writing parameters to a Vertica block.

Inheritance diagram for com.vertica.sdk.ParamWriter:



Collaboration diagram for com.vertica.sdk.ParamWriter:



Public Member Functions

- void addCol (ByteBuffer arg, int colstride, VerticaType dt, String colName)
- void addCol (ByteBuffer arg, int colstride, VerticaType dt)
- void addCol (ByteBuffer arg, ByteBuffer da, int colstride, VerticaType dt)
- void addCol (ByteBuffer arg, ByteBuffer da, int colstride, VerticaType dt, String colName)
- boolean containsParameter (String paramName)

Function to see if the ParamReader has a value for the parameter.

- boolean getBoolean (int idx)

  Get a BOOLEAN value from the input row.
- boolean **getBoolean** (String paramName) throws UdfException
- ByteBuffer getColDataAreaRef (int idx)
- ByteBuffer getColRef (int idx)
- java.sql.Date **getDate** (String paramName) throws UdfException
- java.sql.Date getDate (int idx)

  Get a DATE value from the input row.
- double getDouble (int idx)

  Get a DOUBLE value from the input row.
- double **getDouble** (String paramName) throws UdfException
- int **getIndex** (String paramName) throws UdfException
- long getLong (int idx)

  Get a LONG INTEGER value from the input row.
- long **getLong** (String paramName) throws UdfException
- int getNumCols ()
- int getNumRows ()
- ArrayList< String > getParamNames ()

  Return all names of parameters stored in this ParamReader.
- String **getString** (String paramName) throws UdfException
- String getString (int idx)

  Get a reference to an VARCHAR/CHAR/VARBINARY/BINARY value from the input row.
- int **getStringLength** (String paramName) throws UdfException
- int getStringLength (int idx)

  Get length of the String from the input row.
- int getStringLoc (int idx)

  Get 'location' of the String from the input row.
- java.sql.Timestamp **getTimestamp** (String paramName) throws UdfException
- java.sql.Timestamp getTimestamp (int idx)

  Get a TIMESTAMP value from the input row.
- VerticaType getType (String paramName) throws UdfException

  Return the type of the given parameter.
- SizedColumnTypes getTypeMetaData ()
- VNumeric **getVNumeric** (String paramName) throws UdfException
- VNumeric getVNumeric (int idx)

  Get a reference to a VNumeric value from the input row.
- VString **getVString** (String paramName) throws UdfException
- VString getVString (int idx)

  Get a reference from the input row to an VString value, which represents a SQL VARCHAR/CHAR/VARBINARY/BINARY value.
- boolean isBooleanNull (int idx)

  Check whether a value from the input row is NULL in BOOLEAN type.
- boolean **isBooleanNull** (String paramName) throws UdfException
- boolean **isDateNull** (String paramName) throws UdfException
- boolean isDateNull (int idx)

  Check whether a value from the input row is NULL in DATE type.
- boolean isDoubleNull (int idx)

  Check whether a value from the input row is NULL in DOUBLE type.
- boolean **isDoubleNull** (String paramName) throws UdfException
- boolean isEmpty ()

  Returns true if there are no parameters.

- boolean isLongNull (int idx)

  Check whether a value from the input row is NULL in LONG INTERGER type.
- boolean **isLongNull** (String paramName) throws UdfException
- boolean **isStringNull** (String paramName) throws UdfException
- boolean isStringNull (int idx)

  Check whether a value from the input row is NULL in SQL VARCHAR/CHAR/VARBINARY/BINARY type.
- boolean **isTimestampInfinite** (String paramName) throws UdfException
- boolean isTimestampInfinite (int idx)

  Check whether a TIMESTAMP value from the input row represents 'infinity'.
- boolean **isTimestampInfiniteNeg** (String paramName) throws UdfException
- boolean isTimestampInfiniteNeg (int idx)

  Check whether a TIMESTAMP value from the input row represents '-infinity'.
- boolean **isTimestampInfinitePos** (String paramName) throws UdfException
- boolean isTimestampInfinitePos (int idx)

  Check whether a TIMESTAMP value from the input row represents '+infinity'.
- boolean **isTimestampNull** (String paramName) throws UdfException
- boolean isTimestampNull (int idx)

  Check whether a value from the input row is NULL in TIMESTAMP type.
- boolean next () throws UdfException, DestroyInvocation
- void setBool (String fieldName, boolean r) throws UdfException

  Adds a BOOLEAN value to the output row.
- void setDate (String fieldName, java.sql.Date r) throws UdfException

  Adds a DATE value to the output row.
- void setDouble (String fieldName, double r) throws UdfException

  Adds a FLOAT value to the output row.
- void setLong (String fieldName, Long r) throws UdfException

  Adds a LONG INTEGER value to the output row.
- void setLongString (String fieldName, String r) throws UdfException

  Adds a Long String value to the output row.
- void setNumeric (String fieldName, BigDecimal bd)

  Allocate a new VNumeric object to use as output.
- void setString (String fieldName, String r) throws UdfException

  Adds a String value to the output row.
- void setTimestamp (String fieldName, java.sql.Timestamp r) throws UdfException

  Adds a TIMESTAMP value to the output row.
- void **setTimestampInfiniteNeg** (String fieldName) throws UdfException
- void **setTimestampInfinitePos** (String fieldName) throws UdfException

Public Attributes

- int **count**
- int **index**
- int **ncols**
- HashMap< String, Integer > **paramNameToIndex**
- SizedColumnTypes **typeMetaData**

Protected Member Functions

- void **clear** ()
- void **resetBuffers** ()

Protected Attributes

- ArrayList< ByteBuffer > **coldataareas**
- ArrayList< ByteBuffer > **cols**
- ArrayList< Integer > **colstrides**
- ArrayList< Integer > **currentPos**

Detailed Description

Iterator interface for writing parameters to a Vertica block.

Member Function Documentation

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg,* int *colstride,* VerticaType *dt,* String *colName*) [inherited]

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
colstride	The stride between data instances.
dt	The type of input.
colName	Name of the column

Referenced by com.vertica.sdk.VerticaBlock.addCol().

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg,* int *colstride,* VerticaType *dt*) [inherited]

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
colstride	The stride between data instances.
dt	The type of input.

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg,* ByteBuffer *da,* int *colstride,* VerticaType *dt*) [inherited]

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
da	The location to find out of band string data.
colstride	The stride between data instances.
dt	The type of input.

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg,* ByteBuffer *da,* int *colstride,* VerticaType *dt,* String *colName*) [inherited]

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
da	The location to find out of band string data.
colstride	The stride between data instances.
dt	The type of input.
colName	Name of the column

boolean com.vertica.sdk.BlockReader.getBoolean (int *idx*) `[inherited]`

Get a BOOLEAN value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The value of the idx'th argument, cast as a BOOLEAN.

ByteBuffer com.vertica.sdk.VerticaBlock.getColDataAreaRef (int *idx*) `[inherited]`

Get the ByteBuffer that stores out of line string data (Data Area) for the idx'th argument

Parameters

idx	

Returns

Referenced by com.vertica.sdk.BlockReader.getVString().

ByteBuffer com.vertica.sdk.VerticaBlock.getColRef (int *idx*) `[inherited]`

Returns

a ByteBuffer to the idx'th argument, containing data for the column

Example:

```
* ByteBuffer a = arg_reader.getColPtr(0);
*
```

Referenced by com.vertica.sdk.PartitionWriter.copyFromInput(), com.vertica.sdk.BlockReader.getBoolean(), com.-vertica.sdk.BlockReader.getDouble(), com.vertica.sdk.BlockReader.getLong(), com.vertica.sdk.BlockReader.-getStringLength(), com.vertica.sdk.BlockReader.getStringLoc(), com.vertica.sdk.BlockReader.getVNumeric(), com.vertica.sdk.BlockReader.getVString(), com.vertica.sdk.BlockWriter.getVStringWriter(), com.vertica.sdk.-BlockReader.isBooleanNull(), com.vertica.sdk.ParamWriter.setBool(), com.vertica.sdk.BlockWriter.setBoolean(), com.vertica.sdk.BlockWriter.setBooleanNull(), com.vertica.sdk.ParamWriter.setDouble(), com.vertica.sdk.Block-Writer.setDouble(), com.vertica.sdk.BlockWriter.setDoubleNull(), com.vertica.sdk.PartitionWriter.setLong(), com.-vertica.sdk.BlockWriter.setLongNull(), com.vertica.sdk.ParamWriter.setLongString(), com.vertica.sdk.BlockWriter.-setNumeric(), com.vertica.sdk.ParamWriter.setNumeric(), com.vertica.sdk.BlockWriter.setString(), com.vertica.-sdk.ParamWriter.setString(), and com.vertica.sdk.BlockWriter.setStringNull().

java.sql.Date com.vertica.sdk.BlockReader.getDate (int *idx*) `[inherited]`

Get a DATE value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The value of the idx'th argument, cast as a DATE; null if the column is NULL.

double com.vertica.sdk.BlockReader.getDouble (int *idx*) `[inherited]`

Get a DOUBLE value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The value of the idx'th argument, cast as a DOUBLE.

long com.vertica.sdk.BlockReader.getLong (int *idx*) `[inherited]`

Get a LONG INTEGER value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The value of the idx'th argument, cast as a LONG INTEGER.

Example:

```
* long a = arg_reader.getLong(0);
*
```

Referenced by com.vertica.sdk.BlockReader.getDate(), com.vertica.sdk.BlockReader.getTimestamp(), com.vertica.sdk.BlockReader.isDoubleNull(), com.vertica.sdk.BlockReader.isLongNull(), com.vertica.sdk.BlockReader.isTimestampInfiniteNeg(), and com.vertica.sdk.BlockReader.isTimestampInfinitePos().

int com.vertica.sdk.VerticaBlock.getNumCols ( ) `[inherited]`

Returns

the number of arguments held by this reader.

int com.vertica.sdk.VerticaBlock.getNumRows ( ) `[inherited]`

Returns

the number of rows held by this block.

String com.vertica.sdk.BlockReader.getString (int *idx*) `[inherited]`

Get a reference to an VARCHAR/CHAR/VARBINARY/BINARY value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

a reference to the idx'th argument, cast as an String.

int com.vertica.sdk.BlockReader.getStringLength (int *idx*) `[inherited]`

Get length of the String from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The length of the String in specified column.

Referenced by com.vertica.sdk.BlockReader.getVString(), and com.vertica.sdk.BlockReader.isStringNull().

int com.vertica.sdk.BlockReader.getStringLoc (int *idx*) `[inherited]`

Get 'location' of the String from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The location of the String in specified column. If zero, data is inlined immediately after the header, otherwise data is at offset loc within the data area.

Referenced by com.vertica.sdk.BlockReader.getVString().

java.sql.Timestamp com.vertica.sdk.BlockReader.getTimestamp (int *idx*) `[inherited]`

Get a TIMESTAMP value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The value of the idx'th argument, cast as a TIMESTAMP; null if the column is NULL or represents 'infinity'.

VerticaType com.vertica.sdk.ParamReader.getType (String *paramName*) throws UdfException `[inherited]`

Return the type of the given parameter.

Exceptions

UdfException	

SizedColumnTypes com.vertica.sdk.VerticaBlock.getTypeMetaData ( ) `[inherited]`

Returns

information about the types and numbers of arguments

Referenced by com.vertica.sdk.ParamReader.getType().

VNumeric com.vertica.sdk.BlockReader.getVNumeric (int *idx*) `[inherited]`

Get a reference to a VNumeric value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

A reference to the retrieved value cast as a Numeric.

VString com.vertica.sdk.BlockReader.getVString (int *idx*) `[inherited]`

Get a reference from the input row to an VString value, which represents a SQL VARCHAR/CHAR/VARBINARY/B-INARY value.

Parameters

idx	The column number to retrieve from the input row.

Returns

a reference to the idx'th argument, cast as an VString.

Referenced by com.vertica.sdk.BlockReader.getString().

boolean com.vertica.sdk.BlockReader.isBooleanNull (int *idx*) `[inherited]`

Check whether a value from the input row is NULL in BOOLEAN type.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the value is NULL, false otherwise.

boolean com.vertica.sdk.BlockReader.isDateNull (int *idx*) `[inherited]`

Check whether a value from the input row is NULL in DATE type.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the value is NULL, false otherwise.

Referenced by com.vertica.sdk.BlockReader.getDate().

boolean com.vertica.sdk.BlockReader.isDoubleNull (int *idx*) [inherited]

Check whether a value from the input row is NULL in DOUBLE type.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the value is NULL, false otherwise.

boolean com.vertica.sdk.BlockReader.isLongNull (int *idx*) [inherited]

Check whether a value from the input row is NULL in LONG INTERGER type.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the value is NULL, false otherwise.

Referenced by com.vertica.sdk.BlockReader.isDateNull(), and com.vertica.sdk.BlockReader.isTimestampNull().

boolean com.vertica.sdk.BlockReader.isStringNull (int *idx*) [inherited]

Check whether a value from the input row is NULL in SQL VARCHAR/CHAR/VARBINARY/BINARY type.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the value is NULL, false otherwise.

Referenced by com.vertica.sdk.BlockReader.getString().

boolean com.vertica.sdk.BlockReader.isTimestampInfinite (int *idx*) [inherited]

Check whether a TIMESTAMP value from the input row represents 'infinity'.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the TIMESTAMP value is '+infinity' or '-infinity', false otherwise.

Referenced by com.vertica.sdk.BlockReader.getTimestamp().

boolean com.vertica.sdk.BlockReader.isTimestampInfiniteNeg (int *idx*) `[inherited]`

Check whether a TIMESTAMP value from the input row represents '-infinity'.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the TIMESTAMP value is '-infinity', false otherwise.

Referenced by com.vertica.sdk.BlockReader.isTimestampInfinite().

boolean com.vertica.sdk.BlockReader.isTimestampInfinitePos (int *idx*) `[inherited]`

Check whether a TIMESTAMP value from the input row represents '+infinity'.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the TIMESTAMP value is '+infinity', false otherwise.

Referenced by com.vertica.sdk.BlockReader.isTimestampInfinite().

boolean com.vertica.sdk.BlockReader.isTimestampNull (int *idx*) `[inherited]`

Check whether a value from the input row is NULL in TIMESTAMP type.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the value is NULL, false otherwise.

Referenced by com.vertica.sdk.BlockReader.getTimestamp().

boolean com.vertica.sdk.BlockReader.next ( ) throws UdfException, DestroyInvocation `[inherited]`

Advance to the next record.

Returns

true if there are more rows to read, false otherwise.

void com.vertica.sdk.ParamWriter.setBool (String *fieldName,* boolean *r*) throws UdfException

Adds a BOOLEAN value to the output row.

Parameters

r	The BOOLEAN value to insert into the output row.

void com.vertica.sdk.ParamWriter.setDate (String *fieldName,* java.sql.Date *r*) throws UdfException

Adds a DATE value to the output row.

Parameters

r	The DATE value to insert into the output row.

void com.vertica.sdk.ParamWriter.setDouble (String *fieldName,* double *r*) throws UdfException

Adds a FLOAT value to the output row.

Parameters

r	The FLOAT value to insert into the output row.

void com.vertica.sdk.ParamWriter.setLong (String *fieldName,* Long *r*) throws UdfException

Adds a LONG INTEGER value to the output row.

Setter methods

Parameters

r	The LONG INTEGER value to insert into the output row.

Referenced by com.vertica.sdk.ParamWriter.setDate(), and com.vertica.sdk.ParamWriter.setTimestamp().

void com.vertica.sdk.ParamWriter.setLongString (String *fieldName,* String *r*) throws UdfException

Adds a Long String value to the output row.

Parameters

r	The Long String value to insert into the output row.

void com.vertica.sdk.ParamWriter.setNumeric (String *fieldName,* BigDecimal *bd*)

Allocate a new VNumeric object to use as output.

Returns

A new VNumeric object to hold output. This object automatically added to the output row.

void com.vertica.sdk.ParamWriter.setString (String *fieldName,* String *r*) throws UdfException

Adds a String value to the output row.

Parameters

r	The String value to insert into the output row.

void com.vertica.sdk.ParamWriter.setTimestamp (String *fieldName,* java.sql.Timestamp *r*) throws UdfException

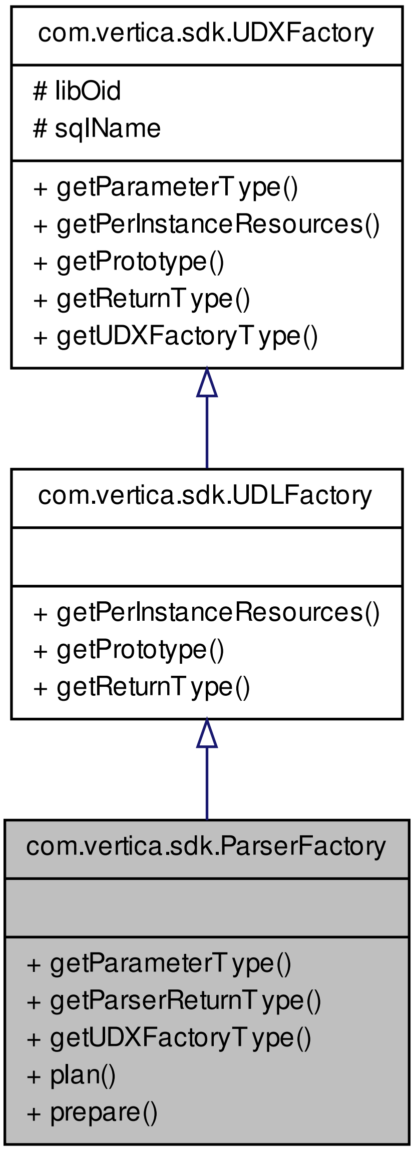
Adds a TIMESTAMP value to the output row.

Parameters

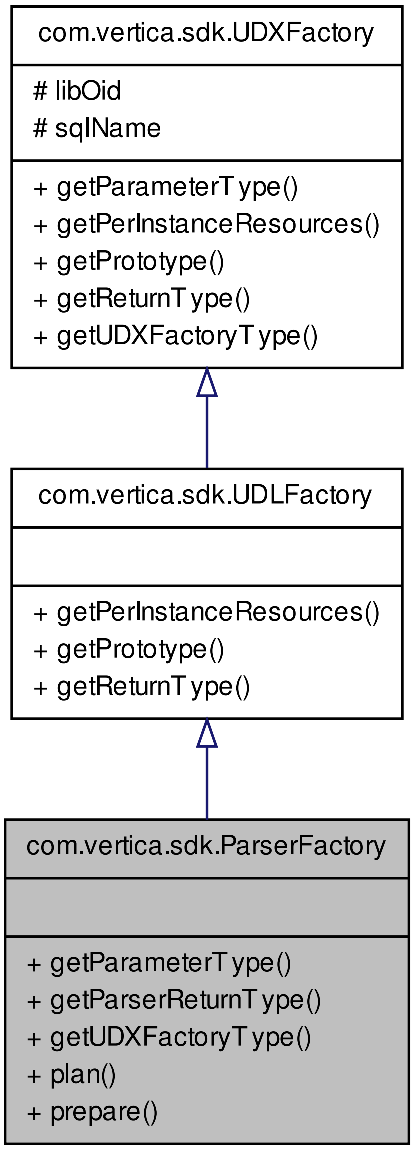
r	The TIMESTAMP value to insert into the output row.

com.vertica.sdk.ParserFactory Class Reference

Inheritance diagram for com.vertica.sdk.ParserFactory:



Collaboration diagram for com.vertica.sdk.ParserFactory:



Public Member Functions

- void getParameterType (ServerInterface srvInterface, SizedColumnTypes parameterTypes)
- void getParserReturnType (ServerInterface srvInterface, PerColumnParamReader perColumnParamReader, PlanContext planCtxt, SizedColumnTypes argTypes, SizedColumnTypes returnType) throws UdfException
- void **getPerInstanceResources** (ServerInterface srvInterface, VResources res)
- void getPrototype (ServerInterface srvInterface, ColumnTypes argTypes, ColumnTypes returnType)
- void getReturnType (ServerInterface srvInterface, SizedColumnTypes argTypes, SizedColumnTypes returnType)
- UDXType getUDXFactoryType ()

- void plan (ServerInterface srvInterface, PerColumnParamReader perColumnParamReader, PlanContext planCtxt) throws UdfException
- abstract UDParser prepare (ServerInterface srvInterface, PerColumnParamReader perColumnParamReader, PlanContext planCtxt, SizedColumnTypes returnType) throws UdfException

Protected Attributes

- long **libOid**
- String **sqlName**

Detailed Description

Construct a single Parser.

Note that ParserFactories are singletons. Subclasses should be stateless, with no fields containing data, just methods. plan() and prepare() methods must never modify any global variables or state; they may only modify the variables that they are given as arguments.

Member Function Documentation

void com.vertica.sdk.ParserFactory.getParameterType (**ServerInterface** *srvInterface,* **SizedColumnTypes** *parameterTypes*)

Inherited from the parent "UDXFactory" class in VerticaUDx.h

void com.vertica.sdk.ParserFactory.getParserReturnType (**ServerInterface** *srvInterface,* **PerColumnParamReader** *perColumnParamReader,* **PlanContext** *planCtxt,* **SizedColumnTypes** *argTypes,* **SizedColumnTypes** *returnType*) throws **UdfException**

Function to tell Vertica what the return types (and length/precision if necessary) of this UDX are. Called, possibly multiple times, on each node executing the query.

The default provided implementation configures Vertica to use the same output column types as the destination table. This requires that the UDParser validate the expected output column types and emit appropriate tuples. Note that the default provided implementation of this function should be sufficient for most Parsers, so this method should not be overridden by most Parser implementations. If a COPY statement has a return type that doesn't match the destination table, Vertica will emit an appropriate error. Users can use COPY expressions to perform typecasting and conversion if necessary.

For CHAR/VARCHAR types, specify the max length,

For Time/Timestamp types (with or without time zone), specify the precision, -1 means unspecified/don't care

For all other types, no length/precision specification needed

Parameters

srvInterface	Interface to server operations and functionality, including (not-per-column) parameter lookup
perColumn-ParamReader	Per-column parameters passed into the query
planCtxt	Context for storing and retrieving arbitrary data, for use just by this instance of this query. The same context is shared with plan().
argTypes	Provides the data types of arguments that this UDT was called with. This may be used to modify the return types accordingly.

returnType	User code must fill in the names and data types returned by the UDT.

Exceptions

UdfException	

void com.vertica.sdk.UDLFactory.getPrototype (ServerInterface *srvInterface*, ColumnTypes *argTypes*, ColumnTypes *returnType*) [virtual], [inherited]

Provides the argument and return types of the UDL. UDL's take no input tuples; as such, their prototype is empty.

Implements com.vertica.sdk.UDXFactory.

void com.vertica.sdk.UDLFactory.getReturnType (ServerInterface *srvInterface*, SizedColumnTypes *argTypes*, SizedColumnTypes *returnType*) [virtual], [inherited]

Not used in this form

Implements com.vertica.sdk.UDXFactory.

UDXType com.vertica.sdk.ParserFactory.getUDXFactoryType ( ) [virtual]

Returns

the type of UDX Object instance this factory returns.

Note

User subclasses should use the appropriate subclass of UDXFactory and not override this method on their own.

Implements com.vertica.sdk.UDXFactory.

void com.vertica.sdk.ParserFactory.plan (ServerInterface *srvInterface*, PerColumnParamReader *perColumnParamReader*, PlanContext *planCtxt*) throws UdfException

Execute any planning logic required at query plan time. This method is run once per query, during query initialization. Its job is to perform parameter validation, and to modify the set of nodes that the COPY statement will run on (through srvInterface).

plan() runs exactly once per query, on the initiator node. If it throws an exception, the query will not proceed; it will be aborted prior to distributing the query to the other nodes and running prepare().

Parameters

srvInterface	Interface to server operations and functionality, including (not-per-column) parameter lookup
perColumn-ParamReader	Per-column parameters passed into the query
planCtxt	Context for storing and retrieving arbitrary data, for use just by this instance of this query. The same context is shared with plan().

Exceptions

UdfException	

abstract UDParser com.vertica.sdk.ParserFactory.prepare (ServerInterface *srvInterface,* PerColumnParamReader *perColumnParamReader,* PlanContext *planCtxt,* SizedColumnTypes *returnType*) throws UdfException `[pure virtual]`

Instantiate a UDParser instance. This function will be called on each node, prior to the Load operator starting to execute.

Parameters

srvInterface	Interface to server operations and functionality, including (not-per-column) parameter lookup
perColumn-ParamReader	Per-column parameters passed into the query
planCtxt	Context for storing and retrieving arbitrary data, for use just by this instance of this query. The same context is shared with plan().
returnType	The data types of the columns that this Parser must produce

Returns

The UDParser instance to be used by this query

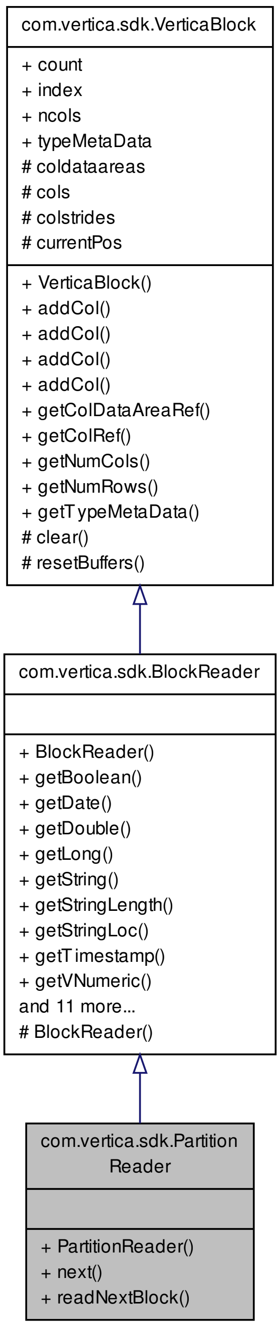
Exceptions

UdfException	

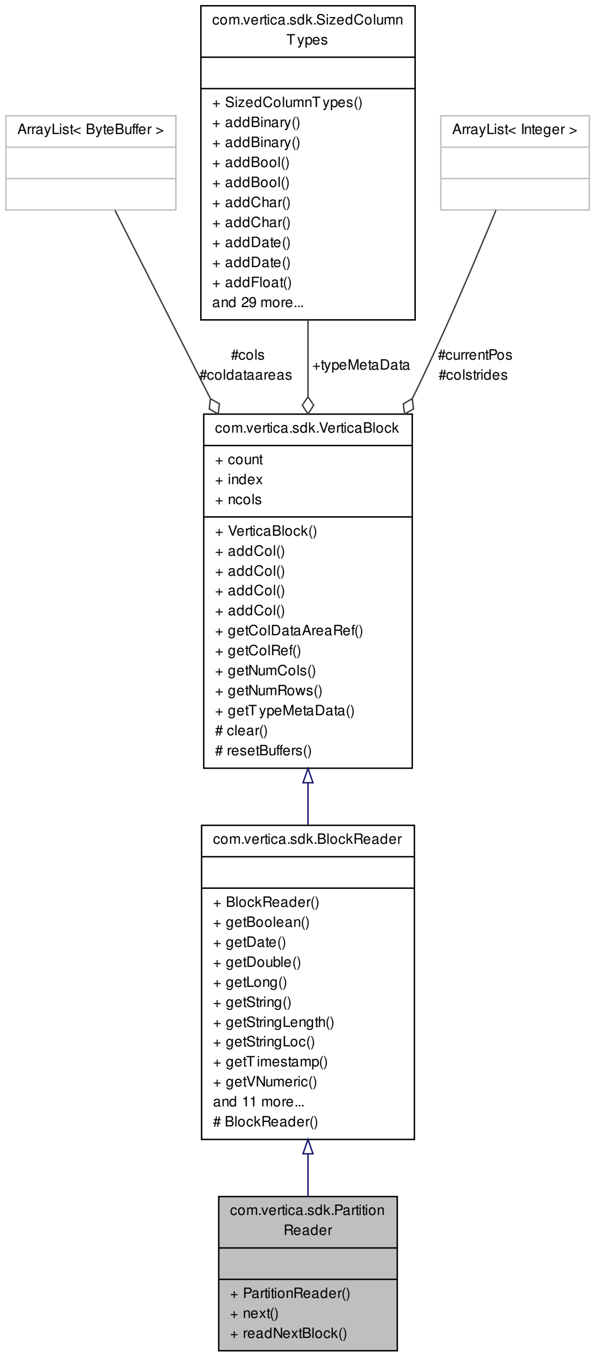
com.vertica.sdk.PartitionReader Class Reference

PartitionReader provides an iterator-based read interface over all input data in a single partition. Automatically fetches data a block-at-a-time, as needed.

Inheritance diagram for com.vertica.sdk.PartitionReader:



Collaboration diagram for com.vertica.sdk.PartitionReader:



Public Member Functions

- void addCol (ByteBuffer arg, int colstride, VerticaType dt, String colName)
- void addCol (ByteBuffer arg, int colstride, VerticaType dt)
- void addCol (ByteBuffer arg, ByteBuffer da, int colstride, VerticaType dt)
- void addCol (ByteBuffer arg, ByteBuffer da, int colstride, VerticaType dt, String colName)
- boolean getBoolean (int idx)

Get a BOOLEAN value from the input row.

- ByteBuffer getColDataAreaRef (int idx)
- ByteBuffer getColRef (int idx)
- java.sql.Date getDate (int idx)

  Get a DATE value from the input row.
- double getDouble (int idx)

  Get a DOUBLE value from the input row.
- long getLong (int idx)

  Get a LONG INTEGER value from the input row.
- int getNumCols ()
- int getNumRows ()
- String getString (int idx)

  Get a reference to an VARCHAR/CHAR/VARBINARY/BINARY value from the input row.
- int getStringLength (int idx)

  Get length of the String from the input row.
- int getStringLoc (int idx)

  Get 'location' of the String from the input row.
- java.sql.Timestamp getTimestamp (int idx)

  Get a TIMESTAMP value from the input row.
- SizedColumnTypes getTypeMetaData ()
- VNumeric getVNumeric (int idx)

  Get a reference to a VNumeric value from the input row.
- VString getVString (int idx)

  Get a reference from the input row to an VString value, which represents a SQL VARCHAR/CHAR/VARBINARY/BINARY value.
- boolean isBooleanNull (int idx)

  Check whether a value from the input row is NULL in BOOLEAN type.
- boolean isDateNull (int idx)

  Check whether a value from the input row is NULL in DATE type.
- boolean isDoubleNull (int idx)

  Check whether a value from the input row is NULL in DOUBLE type.
- boolean isLongNull (int idx)

  Check whether a value from the input row is NULL in LONG INTERGER type.
- boolean isStringNull (int idx)

  Check whether a value from the input row is NULL in SQL VARCHAR/CHAR/VARBINARY/BINARY type.
- boolean isTimestampInfinite (int idx)

  Check whether a TIMESTAMP value from the input row represents 'infinity'.
- boolean isTimestampInfiniteNeg (int idx)

  Check whether a TIMESTAMP value from the input row represents '-infinity'.
- boolean isTimestampInfinitePos (int idx)

  Check whether a TIMESTAMP value from the input row represents '+infinity'.
- boolean isTimestampNull (int idx)

  Check whether a value from the input row is NULL in TIMESTAMP type.
- boolean **next** () throws UdfException, DestroyInvocation
- abstract boolean readNextBlock () throws UdfException, DestroyInvocation

Public Attributes

- int **count**
- int **index**
- int **ncols**
- SizedColumnTypes **typeMetaData**

Protected Member Functions

- void **clear** ()
- void **resetBuffers** ()

Protected Attributes

- ArrayList< ByteBuffer > **coldataareas**
- ArrayList< ByteBuffer > **cols**
- ArrayList< Integer > **colstrides**
- ArrayList< Integer > **currentPos**

Detailed Description

PartitionReader provides an iterator-based read interface over all input data in a single partition. Automatically fetches data a block-at-a-time, as needed.

Member Function Documentation

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, int *colstride*, VerticaType *dt*, String *colName*) `[inherited]`

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
colstride	The stride between data instances.
dt	The type of input.
colName	Name of the column

Referenced by com.vertica.sdk.VerticaBlock.addCol().

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, int *colstride*, VerticaType *dt*) `[inherited]`

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
colstride	The stride between data instances.
dt	The type of input.

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, ByteBuffer *da*, int *colstride*, VerticaType *dt*) `[inherited]`

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
da	The location to find out of band string data.

colstride	The stride between data instances.
dt	The type of input.

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, ByteBuffer *da*, int *colstride*, VerticaType *dt*, String *colName*) `[inherited]`

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
da	The location to find out of band string data.
colstride	The stride between data instances.
dt	The type of input.
colName	Name of the column

boolean com.vertica.sdk.BlockReader.getBoolean (int *idx*) `[inherited]`

Get a BOOLEAN value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The value of the idx'th argument, cast as a BOOLEAN.

ByteBuffer com.vertica.sdk.VerticaBlock.getColDataAreaRef (int *idx*) `[inherited]`

Get the ByteBuffer that stores out of line string data (Data Area) for the idx'th argument

Parameters

idx	

Returns

Referenced by com.vertica.sdk.BlockReader.getVString().

ByteBuffer com.vertica.sdk.VerticaBlock.getColRef (int *idx*) `[inherited]`

Returns

a ByteBuffer to the idx'th argument, containing data for the column

Example:

```
* ByteBuffer a = arg_reader.getColPtr(0);
*
```

Referenced by com.vertica.sdk.PartitionWriter.copyFromInput(), com.vertica.sdk.BlockReader.getBoolean(), com.-vertica.sdk.BlockReader.getDouble(), com.vertica.sdk.BlockReader.getLong(), com.vertica.sdk.BlockReader.-getStringLength(), com.vertica.sdk.BlockReader.getStringLoc(), com.vertica.sdk.BlockReader.getVNumeric(), com.vertica.sdk.BlockReader.getVString(), com.vertica.sdk.BlockWriter.getVStringWriter(), com.vertica.sdk.-BlockReader.isBooleanNull(), com.vertica.sdk.ParamWriter.setBool(), com.vertica.sdk.BlockWriter.setBoolean(), com.vertica.sdk.BlockWriter.setBooleanNull(), com.vertica.sdk.ParamWriter.setDouble(), com.vertica.sdk.Block-Writer.setDouble(), com.vertica.sdk.BlockWriter.setDoubleNull(), com.vertica.sdk.PartitionWriter.setLong(), com.-vertica.sdk.BlockWriter.setLongNull(), com.vertica.sdk.ParamWriter.setLongString(), com.vertica.sdk.BlockWriter.-setNumeric(), com.vertica.sdk.ParamWriter.setNumeric(), com.vertica.sdk.BlockWriter.setString(), com.vertica.-sdk.ParamWriter.setString(), and com.vertica.sdk.BlockWriter.setStringNull().

java.sql.Date com.vertica.sdk.BlockReader.getDate (int *idx*) `[inherited]`

Get a DATE value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The value of the idx'th argument, cast as a DATE; null if the column is NULL.

double com.vertica.sdk.BlockReader.getDouble (int *idx*) `[inherited]`

Get a DOUBLE value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The value of the idx'th argument, cast as a DOUBLE.

long com.vertica.sdk.BlockReader.getLong (int *idx*) `[inherited]`

Get a LONG INTEGER value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The value of the idx'th argument, cast as a LONG INTEGER.

Example:

```
* long a = arg_reader.getLong(0);
*
```

Referenced by com.vertica.sdk.BlockReader.getDate(), com.vertica.sdk.BlockReader.getTimestamp(), com.-vertica.sdk.BlockReader.isDoubleNull(), com.vertica.sdk.BlockReader.isLongNull(), com.vertica.sdk.BlockReader.-isTimestampInfiniteNeg(), and com.vertica.sdk.BlockReader.isTimestampInfinitePos().

int com.vertica.sdk.VerticaBlock.getNumCols ( ) `[inherited]`

Returns

the number of arguments held by this reader.

int com.vertica.sdk.VerticaBlock.getNumRows ( ) `[inherited]`

Returns

the number of rows held by this block.

String com.vertica.sdk.BlockReader.getString (int *idx*) `[inherited]`

Get a reference to an VARCHAR/CHAR/VARBINARY/BINARY value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

a reference to the idx'th argument, cast as an String.

int com.vertica.sdk.BlockReader.getStringLength (int *idx*) `[inherited]`

Get length of the String from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The length of the String in specified column.

Referenced by com.vertica.sdk.BlockReader.getVString(), and com.vertica.sdk.BlockReader.isStringNull().

int com.vertica.sdk.BlockReader.getStringLoc (int *idx*) `[inherited]`

Get 'location' of the String from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The location of the String in specified column. If zero, data is inlined immediately after the header, otherwise data is at offset loc within the data area.

Referenced by com.vertica.sdk.BlockReader.getVString().

java.sql.Timestamp com.vertica.sdk.BlockReader.getTimestamp (int *idx*) `[inherited]`

Get a TIMESTAMP value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

The value of the idx'th argument, cast as a TIMESTAMP; null if the column is NULL or represents 'infinity'.

SizedColumnTypes com.vertica.sdk.VerticaBlock.getTypeMetaData ( ) `[inherited]`

Returns

information about the types and numbers of arguments

Referenced by com.vertica.sdk.ParamReader.getType().

VNumeric com.vertica.sdk.BlockReader.getVNumeric (int *idx*) `[inherited]`

Get a reference to a VNumeric value from the input row.

Parameters

idx	The column number to retrieve from the input row.

Returns

A reference to the retrieved value cast as a Numeric.

VString com.vertica.sdk.BlockReader.getVString (int *idx*) `[inherited]`

Get a reference from the input row to an VString value, which represents a SQL VARCHAR/CHAR/VARBINARY/B-INARY value.

Parameters

idx	The column number to retrieve from the input row.

Returns

a reference to the idx'th argument, cast as an VString.

Referenced by com.vertica.sdk.BlockReader.getString().

boolean com.vertica.sdk.BlockReader.isBooleanNull (int *idx*) `[inherited]`

Check whether a value from the input row is NULL in BOOLEAN type.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the value is NULL, false otherwise.

boolean com.vertica.sdk.BlockReader.isDateNull (int *idx*) `[inherited]`

Check whether a value from the input row is NULL in DATE type.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the value is NULL, false otherwise.

Referenced by com.vertica.sdk.BlockReader.getDate().

boolean com.vertica.sdk.BlockReader.isDoubleNull (int *idx*) `[inherited]`

Check whether a value from the input row is NULL in DOUBLE type.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the value is NULL, false otherwise.

boolean com.vertica.sdk.BlockReader.isLongNull (int *idx*) `[inherited]`

Check whether a value from the input row is NULL in LONG INTERGER type.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the value is NULL, false otherwise.

Referenced by com.vertica.sdk.BlockReader.isDateNull(), and com.vertica.sdk.BlockReader.isTimestampNull().

boolean com.vertica.sdk.BlockReader.isStringNull (int *idx*) `[inherited]`

Check whether a value from the input row is NULL in SQL VARCHAR/CHAR/VARBINARY/BINARY type.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the value is NULL, false otherwise.

Referenced by com.vertica.sdk.BlockReader.getString().

boolean com.vertica.sdk.BlockReader.isTimestampInfinite (int *idx*) `[inherited]`

Check whether a TIMESTAMP value from the input row represents ’infinity’.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the TIMESTAMP value is '+infinity' or '-infinity', false otherwise.

Referenced by com.vertica.sdk.BlockReader.getTimestamp().

boolean com.vertica.sdk.BlockReader.isTimestampInfiniteNeg (int *idx*) `[inherited]`

Check whether a TIMESTAMP value from the input row represents '-infinity'.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the TIMESTAMP value is '-infinity', false otherwise.

Referenced by com.vertica.sdk.BlockReader.isTimestampInfinite().

boolean com.vertica.sdk.BlockReader.isTimestampInfinitePos (int *idx*) `[inherited]`

Check whether a TIMESTAMP value from the input row represents '+infinity'.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the TIMESTAMP value is '+infinity', false otherwise.

Referenced by com.vertica.sdk.BlockReader.isTimestampInfinite().

boolean com.vertica.sdk.BlockReader.isTimestampNull (int *idx*) `[inherited]`

Check whether a value from the input row is NULL in TIMESTAMP type.

Parameters

idx	The column number to retrieve from the input row.

Returns

true if the value is NULL, false otherwise.

Referenced by com.vertica.sdk.BlockReader.getTimestamp().

abstract boolean com.vertica.sdk.PartitionReader.readNextBlock ( ) throws UdfException, DestroyInvocation `[pure virtual]`

Reads in the next block of data and positions cursor at the beginning.

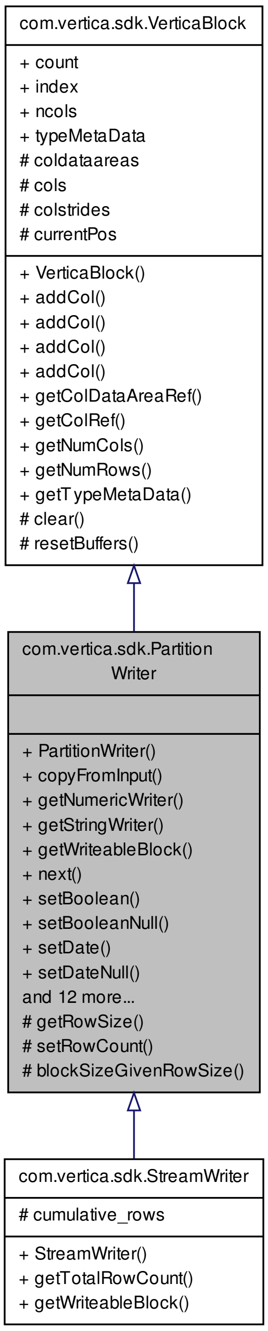
Returns

false if there's no more input data

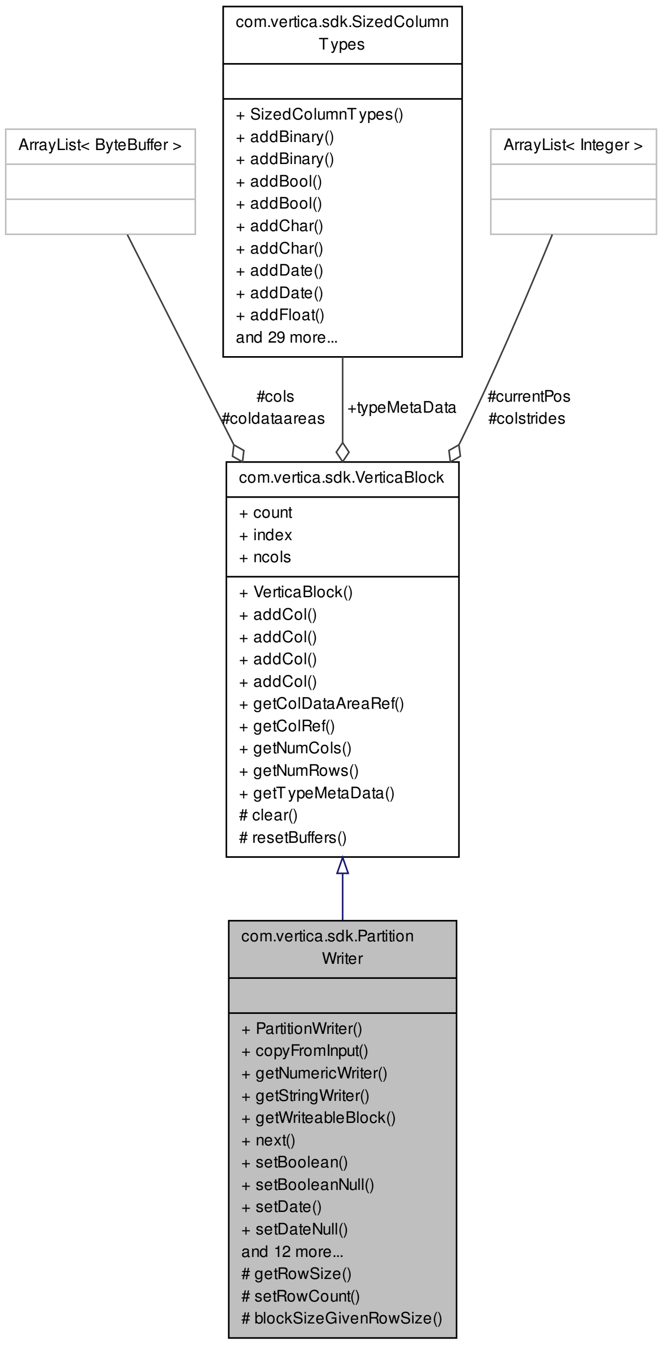
com.vertica.sdk.PartitionWriter Class Reference

PartitionWriter provides an iterator-based write interface over output data for a single partition. Automatically makes space a block-at-a-time, as needed.

Inheritance diagram for com.vertica.sdk.PartitionWriter:



Collaboration diagram for com.vertica.sdk.PartitionWriter:



Public Member Functions

- **PartitionWriter** (int nargs)
- void addCol (ByteBuffer arg, int colstride, VerticaType dt, String colName)
- void addCol (ByteBuffer arg, int colstride, VerticaType dt)
- void addCol (ByteBuffer arg, ByteBuffer da, int colstride, VerticaType dt)
- void addCol (ByteBuffer arg, ByteBuffer da, int colstride, VerticaType dt, String colName)

- void copyFromInput (int dstIdx, PartitionReader input_reader, int srcIdx) throws UdfException
- ByteBuffer getColDataAreaRef (int idx)
- ByteBuffer getColRef (int idx)
- int getNumCols ()
- VNumeric **getNumericWriter** (int idx)
- int getNumRows ()
- VString **getStringWriter** (int idx)
- SizedColumnTypes getTypeMetaData ()
- abstract boolean getWriteableBlock () throws UdfException, DestroyInvocation
- boolean **next** () throws UdfException, DestroyInvocation
- void **setBoolean** (int idx, boolean r)
- void **setBooleanNull** (int idx)
- void **setDate** (int idx, java.sql.Date r)
- void **setDateNull** (int idx)
- void **setDouble** (int idx, double r)
- void **setDoubleNull** (int idx)
- void setLong (int idx, long r)
- void **setLongNull** (int idx)
- void **setNumeric** (int idx, BigDecimal bd)
- void **setNumericNull** (int idx)
- void **setString** (int idx, String r)
- void **setStringNull** (int idx)
- void **setTimestamp** (int idx, java.sql.Timestamp r)
- void **setTimestampInfiniteNeg** (int idx)
- void **setTimestampInfinitePos** (int idx)
- void **setTimestampNull** (int idx)

Public Attributes

- int **count**
- int **index**
- int **ncols**
- SizedColumnTypes **typeMetaData**

Protected Member Functions

- void **clear** ()
- int **getRowSize** (SizedColumnTypes types)
- void **resetBuffers** ()
- void **setRowCount** (SizedColumnTypes types)

Static Protected Member Functions

- static int **blockSizeGivenRowSize** (int row_size)

Protected Attributes

- ArrayList< ByteBuffer > **coldataareas**
- ArrayList< ByteBuffer > **cols**
- ArrayList< Integer > **colstrides**
- ArrayList< Integer > **currentPos**

Detailed Description

PartitionWriter provides an iterator-based write interface over output data for a single partition. Automatically makes space a block-at-a-time, as needed.

Member Function Documentation

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, int *colstride*, VerticaType *dt*, String *colName*) `[inherited]`

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
colstride	The stride between data instances.
dt	The type of input.
colName	Name of the column

Referenced by com.vertica.sdk.VerticaBlock.addCol().

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, int *colstride*, VerticaType *dt*) `[inherited]`

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
colstride	The stride between data instances.
dt	The type of input.

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, ByteBuffer *da*, int *colstride*, VerticaType *dt*) `[inherited]`

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
da	The location to find out of band string data.
colstride	The stride between data instances.
dt	The type of input.

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, ByteBuffer *da*, int *colstride*, VerticaType *dt*, String *colName*) `[inherited]`

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
da	The location to find out of band string data.
colstride	The stride between data instances.

dt	The type of input.
colName	Name of the column

void com.vertica.sdk.PartitionWriter.copyFromInput (int *dstIdx*, PartitionReader *input_reader*, int *srcIdx*) throws UdfException

Copies a column from the input reader to the output writer. The data types and sizes of the source and destination columns must match exactly.

Parameters

dstIdx	The destination column index (in the output writer)
input_reader	The input reader from which to copy a column
srcIdx	The source column index (in the input reader)

ByteBuffer com.vertica.sdk.VerticaBlock.getColDataAreaRef (int *idx*) `[inherited]`

Get the ByteBuffer that stores out of line string data (Data Area) for the idx'th argument

Parameters

idx	

Returns

Referenced by com.vertica.sdk.BlockReader.getVString().

ByteBuffer com.vertica.sdk.VerticaBlock.getColRef (int *idx*) `[inherited]`

Returns

a ByteBuffer to the idx'th argument, containing data for the column

Example:

```
* ByteBuffer a = arg_reader.getColPtr(0);
*
```

Referenced by com.vertica.sdk.PartitionWriter.copyFromInput(), com.vertica.sdk.BlockReader.getBoolean(), com.-vertica.sdk.BlockReader.getDouble(), com.vertica.sdk.BlockReader.getLong(), com.vertica.sdk.BlockReader.-getStringLength(), com.vertica.sdk.BlockReader.getStringLoc(), com.vertica.sdk.BlockReader.getVNumeric(), com.vertica.sdk.BlockReader.getVString(), com.vertica.sdk.BlockWriter.getVStringWriter(), com.vertica.sdk.-BlockReader.isBooleanNull(), com.vertica.sdk.ParamWriter.setBool(), com.vertica.sdk.BlockWriter.setBoolean(), com.vertica.sdk.BlockWriter.setBooleanNull(), com.vertica.sdk.ParamWriter.setDouble(), com.vertica.sdk.Block-Writer.setDouble(), com.vertica.sdk.BlockWriter.setDoubleNull(), com.vertica.sdk.PartitionWriter.setLong(), com.-vertica.sdk.BlockWriter.setLongNull(), com.vertica.sdk.ParamWriter.setLongString(), com.vertica.sdk.BlockWriter.-setNumeric(), com.vertica.sdk.ParamWriter.setNumeric(), com.vertica.sdk.BlockWriter.setString(), com.vertica.-sdk.ParamWriter.setString(), and com.vertica.sdk.BlockWriter.setStringNull().

int com.vertica.sdk.VerticaBlock.getNumCols ( ) `[inherited]`

Returns

the number of arguments held by this reader.

int com.vertica.sdk.VerticaBlock.getNumRows ( ) `[inherited]`

Returns

the number of rows held by this block.

SizedColumnTypes com.vertica.sdk.VerticaBlock.getTypeMetaData ( ) `[inherited]`

Returns

information about the types and numbers of arguments

Referenced by com.vertica.sdk.ParamReader.getType().

abstract boolean com.vertica.sdk.PartitionWriter.getWriteableBlock ( ) throws UdfException, DestroyInvocation `[pure virtual]`

Gets a writeable block of data and positions cursor at the beginning.

Implemented in com.vertica.sdk.StreamWriter.

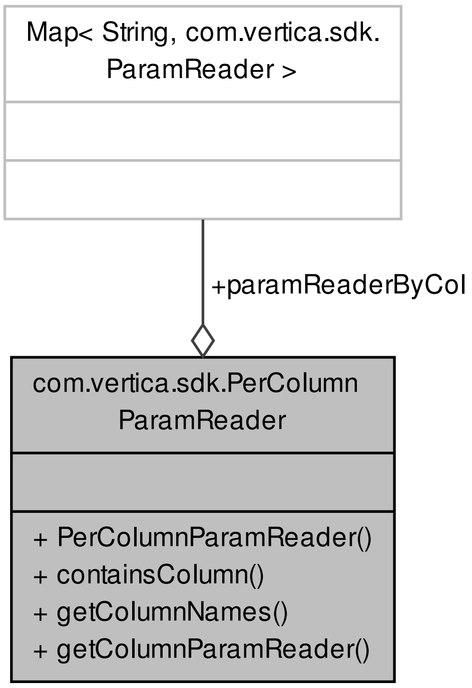
void com.vertica.sdk.PartitionWriter.setLong (int *idx,* long *r*)

Setter methods

com.vertica.sdk.PerColumnParamReader Class Reference

: A wrapper around a map from column to ParamReader.

Collaboration diagram for com.vertica.sdk.PerColumnParamReader:



Public Member Functions

- boolean containsColumn (String columnName)

  Returns true if a ParamReader exists for the given column.
- Collection< String > getColumnNames ()

  Gets the names of all columns with column specific arguments.
- ParamReader getColumnParamReader (String column)

  Gets the parameters of the given column.

Public Attributes

- Map< String, ParamReader > **paramReaderByCol**

Detailed Description

: A wrapper around a map from column to ParamReader.

Member Function Documentation

Collection<String> com.vertica.sdk.PerColumnParamReader.getColumnNames ( )

Gets the names of all columns with column specific arguments.

Returns

a vector of column names

ParamReader com.vertica.sdk.PerColumnParamReader.getColumnParamReader (String *column*)

Gets the parameters of the given column.

Parameters

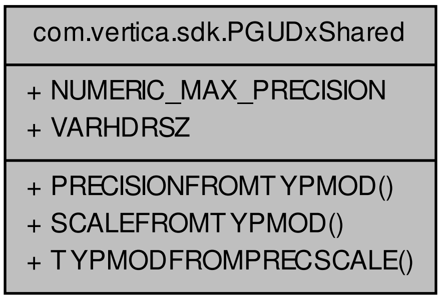
the	name of the column of interest

Returns

the parameters of the given column

com.vertica.sdk.PGUDxShared Class Reference

Collaboration diagram for com.vertica.sdk.PGUDxShared:



Static Public Member Functions

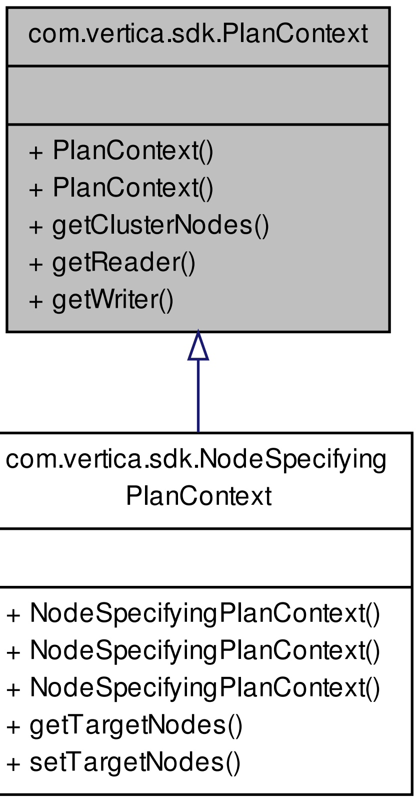
- static long **PRECISIONFROMTYPMOD** (long t)
- static long **SCALEFROMTYPMOD** (long t)
- static long **TYPMODFROMPRECSCALE** (long p, long s)

Static Public Attributes

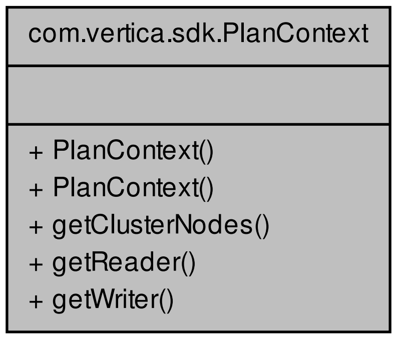
- static final int **NUMERIC_MAX_PRECISION** = 1024
- static final int **VARHDRSZ** = 4

com.vertica.sdk.PlanContext Class Reference

Inheritance diagram for com.vertica.sdk.PlanContext:


com.vertica.sdk.PlanContext
+ PlanContext()
+ PlanContext()
+ getClusterNodes()
+ getReader()
+ getWriter()
com.vertica.sdk.NodeSpecifying PlanContext
+ NodeSpecifyingPlanContext()
+ NodeSpecifyingPlanContext()
+ NodeSpecifyingPlanContext()
+ getTargetNodes()
+ setTargetNodes()

Collaboration diagram for com.vertica.sdk.PlanContext:



Public Member Functions

- **PlanContext** (ParamWriter writer, ArrayList< String > clusterNodes)
- **PlanContext** (ParamWriter writer)
- ArrayList< String > getClusterNodes ()
- ParamReader getReader ()
- ParamWriter getWriter ()

Detailed Description

Interface that allows storage of query-plan state, when different parts of query planning take place on different computers. For example, if some work is done on the query initiator node and some is done on each node executing the query.

Member Function Documentation

ArrayList<String> com.vertica.sdk.PlanContext.getClusterNodes ( )

Get a list of all of the nodes in the current cluster, by node name

ParamReader com.vertica.sdk.PlanContext.getReader ( )

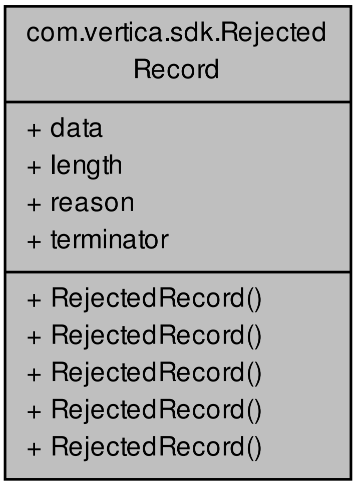
Get a read-only instance of the current context

ParamWriter com.vertica.sdk.PlanContext.getWriter ( )

Get the current context for writing

com.vertica.sdk.RejectedRecord Class Reference

Collaboration diagram for com.vertica.sdk.RejectedRecord:



Public Member Functions

- **RejectedRecord** (String reason, char[ ] data, int length, String terminator)
- **RejectedRecord** (String reason, char[ ] data, int length)
- **RejectedRecord** (String reason, char[ ] data)
- **RejectedRecord** (String reason)

Public Attributes

- char[ ] **data**
- int **length**
- String **reason**
- String **terminator**

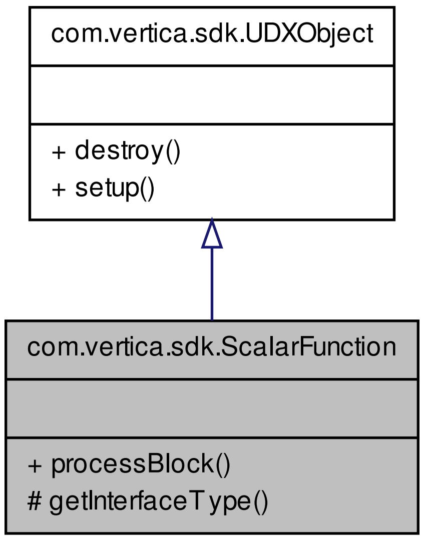
Detailed Description

Information about a rejected record.

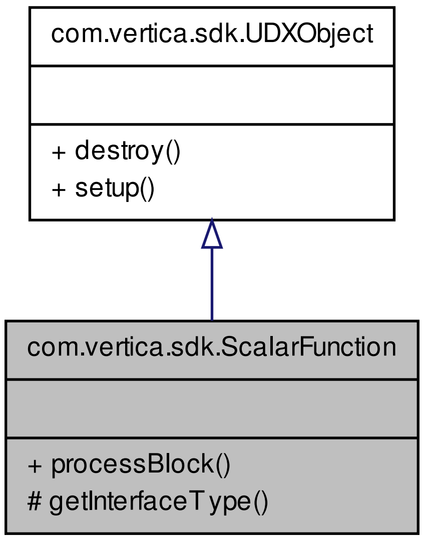
com.vertica.sdk.ScalarFunction Class Reference

Interface for User Defined Scalar Function, the actual code to process a block of data.

Inheritance diagram for com.vertica.sdk.ScalarFunction:



Collaboration diagram for com.vertica.sdk.ScalarFunction:



Classes

- enum InterfaceType

Public Member Functions

- void destroy (ServerInterface srvInterface, SizedColumnTypes argTypes)
- abstract void processBlock (ServerInterface srvInterface, BlockReader arg_reader, BlockWriter res_writer) throws UdfException, DestroyInvocation
- void setup (ServerInterface srvInterface, SizedColumnTypes argTypes)

Protected Member Functions

- InterfaceType **getInterfaceType** ()

Detailed Description

Interface for User Defined Scalar Function, the actual code to process a block of data.

Member Function Documentation

void com.vertica.sdk.UDXObject.destroy (ServerInterface *srvInterface,* SizedColumnTypes *argTypes*) `[inherited]`

Perform per instance destruction. This function may throw errors

abstract void com.vertica.sdk.ScalarFunction.processBlock (ServerInterface *srvInterface,* BlockReader *arg_reader,* BlockWriter *res_writer*) throws UdfException, DestroyInvocation `[pure virtual]`

Invoke a user defined function on a set of rows. As the name suggests, a batch of rows are passed in for every invocation to amortize performance.

Parameters

srvInterface	a ServerInterface object used to communicate with Vertica
arg_reader	input rows
res_writer	output location

Note

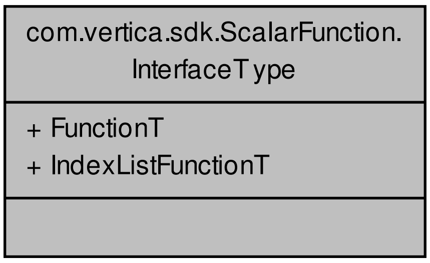
- This methods may be invoked by different threads at different times, and by a different thread than the constructor.
- The order in which the function sees rows is not guaranteed.
- To report error to Vertica, throw a UdfException object

void com.vertica.sdk.UDXObject.setup (ServerInterface *srvInterface,* SizedColumnTypes *argTypes*) `[inherited]`

Perform per instance initialization. This function may throw errors.

com.vertica.sdk.ScalarFunction.InterfaceType Enum Reference

Collaboration diagram for com.vertica.sdk.ScalarFunction.InterfaceType:

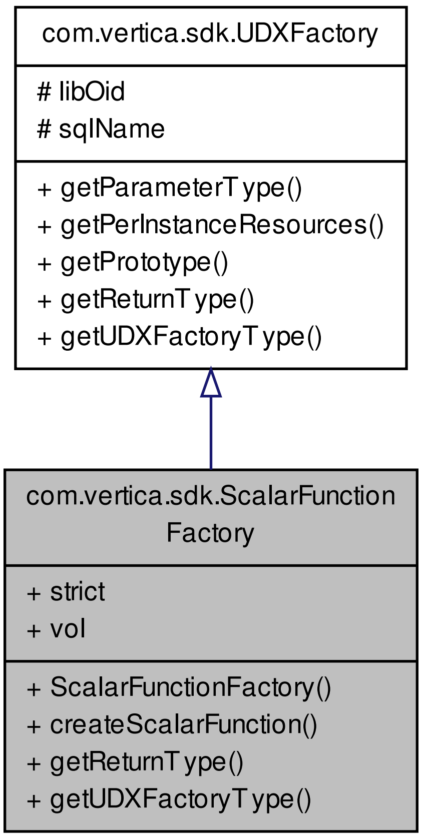


Public Attributes

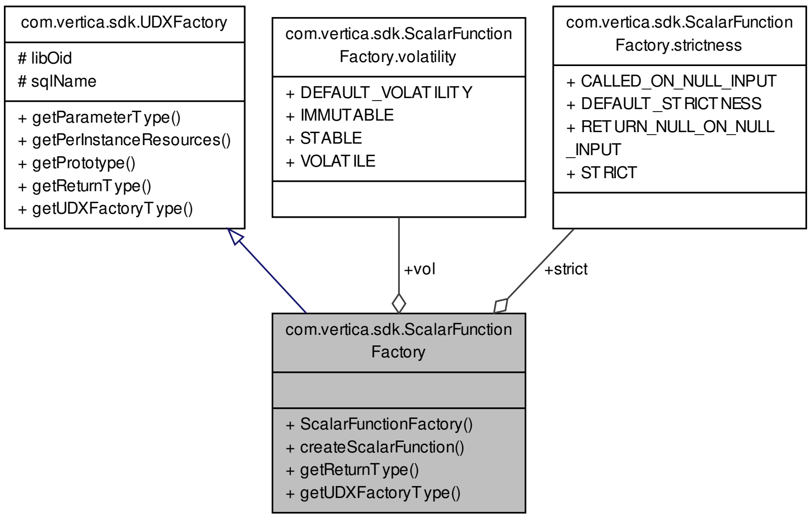
- **FunctionT**
- **IndexListFunctionT**

com.vertica.sdk.ScalarFunctionFactory Class Reference

Inheritance diagram for com.vertica.sdk.ScalarFunctionFactory:



Collaboration diagram for com.vertica.sdk.ScalarFunctionFactory:



Classes

- enum strictness
- enum volatility

Public Member Functions

- abstract ScalarFunction createScalarFunction (ServerInterface srvInterface)
- void getParameterType (ServerInterface srvInterface, SizedColumnTypes parameterTypes)
- void getPerInstanceResources (ServerInterface srvInterface, VResources res)
- abstract void getPrototype (ServerInterface srvInterface, ColumnTypes argTypes, ColumnTypes returnType)
- void getReturnType (ServerInterface srvInterface, SizedColumnTypes argTypes, SizedColumnTypes return-Type) throws UdfException
- UDXType getUDXFactoryType ()

Public Attributes

- strictness **strict**
- volatility vol

Protected Attributes

- long **libOid**
- String **sqlName**

Member Function Documentation

abstract ScalarFunction com.vertica.sdk.ScalarFunctionFactory.createScalarFunction (ServerInterface *srvInterface*)
`[pure virtual]`

Returns

an ScalarFunction object which implements the UDx API described by this metadata.

Parameters

srvInterface	a ServerInterface object used to communicate with Vertica

Note

More than one object may be instantiated per query.

void com.vertica.sdk.UDXFactory.getParameterType (ServerInterface *srvInterface,* SizedColumnTypes *parameterTypes*) [inherited]

Function to tell Vertica the name and types of parameters that this function uses. Vertica will use this to warn function callers that certain parameters they provide are not affecting anything, or that certain parameters that are not being set are reverting to default values.

void com.vertica.sdk.UDXFactory.getPerInstanceResources (ServerInterface *srvInterface,* VResources *res*) [inherited]

Set the resource required for each instance of the UDX Object subclass

Parameters

srvInterface	a ServerInterface object used to communicate with Vertica
res	a VResources object used to tell Vertica what resources are needed by the UDX

abstract void com.vertica.sdk.UDXFactory.getPrototype (ServerInterface *srvInterface,* ColumnTypes *argTypes,* ColumnTypes *returnType*) [pure virtual], [inherited]

Provides the argument and return types of the UDX

Implemented in com.vertica.sdk.UDLFactory.

Referenced by com.vertica.sdk.ScalarFunctionFactory.getReturnType().

void com.vertica.sdk.ScalarFunctionFactory.getReturnType (ServerInterface *srvInterface,* SizedColumnTypes *argTypes,* SizedColumnTypes *returnType*) throws UdfException [virtual]

For scalar functions, this function needs to be overridden only if the return type needs length/precision specification.

Parameters

srvInterface	a ServerInterface object used to communicate with Vertica
argTypes	The data type of the return value defined by processBlock()
returnType	The size of the data returned by processBlock()

Implements com.vertica.sdk.UDXFactory.

UDXType com.vertica.sdk.ScalarFunctionFactory.getUDXFactoryType ( ) [virtual]

Returns

the object type internally used by Vertica

Implements com.vertica.sdk.UDXFactory.

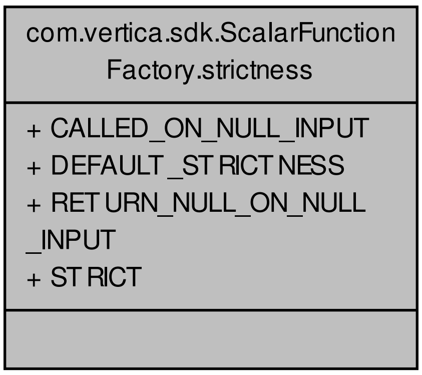
Member Data Documentation

volatility com.vertica.sdk.ScalarFunctionFactory.vol

Strictness and Volatility settings that the UDSF programmer can set Defaults are VOLATILE and CALLED_ON_NULL_INPUT

com.vertica.sdk.ScalarFunctionFactory.strictness Enum Reference

Collaboration diagram for com.vertica.sdk.ScalarFunctionFactory.strictness:

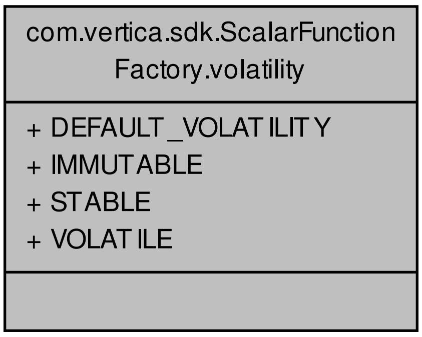

com.vertica.sdk.ScalarFunction Factory.strictness
+ CALLED_ON_NULL_INPUT
+ DEFAULT_STRICTNESS
+ RETURN_NULL_ON_NULL_INPUT
+ STRICT

Public Attributes

- **CALLED_ON_NULL_INPUT**
- **DEFAULT_STRICTNESS**
- **RETURN_NULL_ON_NULL_INPUT**
- **STRICT**

com.vertica.sdk.ScalarFunctionFactory.volatility Enum Reference

Collaboration diagram for com.vertica.sdk.ScalarFunctionFactory.volatility:



Public Attributes

- **DEFAULT_VOLATILITY**
- **IMMUTABLE**
- **STABLE**
- **VOLATILE**

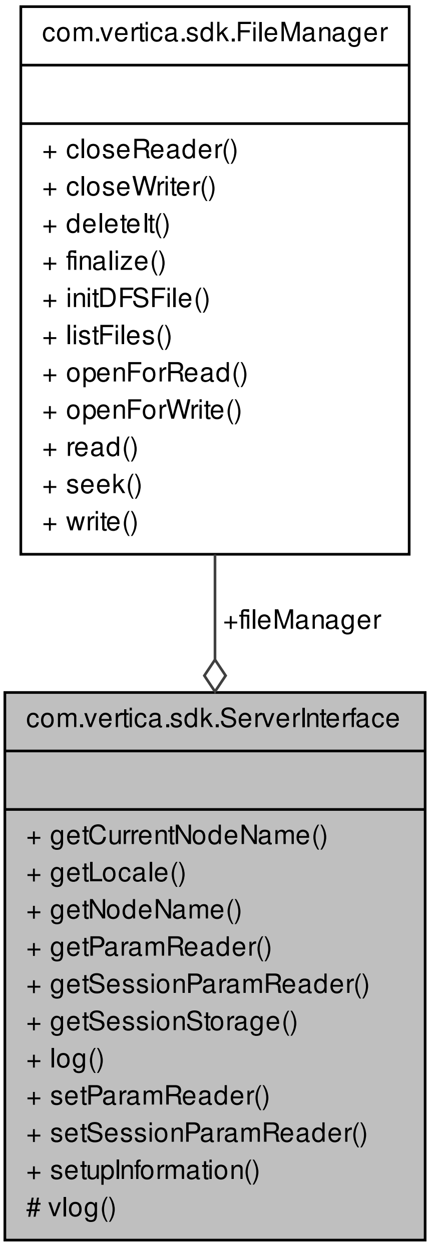
Detailed Description

Enums to allow programmatic specification of volatility and strictness

com.vertica.sdk.ServerInterface Class Reference

Interface that UDX writers can use to interact with the Vertica Server.

Collaboration diagram for com.vertica.sdk.ServerInterface:



Public Member Functions

- String **getCurrentNodeName** ()
- String getLocale ()
- String getNodeName ()
- ParamReader getParamReader ()
- ParamReader getSessionParamReader ()
- Map< String, Object > **getSessionStorage** ()

- void log (String format, Object...args)
- void setParamReader (ParamReader paramReader)
- void setSessionParamReader (ParamReader sessionParamReader)
- void **setupInformation** (String sqlName, ParamReader paramReader, ParamReader sessionParamReader, String locale, String nodeName, FileManager fileManager)

Public Attributes

- FileManager fileManager

Protected Member Functions

- abstract void vlog (String logEntry)

Detailed Description

Interface that UDX writers can use to interact with the Vertica Server.

Member Function Documentation

String com.vertica.sdk.ServerInterface.getLocale ( )

Returns

the locale of the current session.

String com.vertica.sdk.ServerInterface.getNodeName ( )

Returns

the node name of the current session.

ParamReader com.vertica.sdk.ServerInterface.getParamReader ( )

Returns the ParamReader that allows accessing parameter values using their names

ParamReader com.vertica.sdk.ServerInterface.getSessionParamReader ( )

Returns the SessionParamReader that allows accessing session parameter values using their names

void com.vertica.sdk.ServerInterface.log (String *format*, Object... *args*)

Write a message to the vertica.log system log. The message will contain the SQL name of the user defined function or transform being called

Parameters

format	a printf style format string specifying the log message format.

void com.vertica.sdk.ServerInterface.setParamReader (ParamReader *paramReader*)

Set the paramReader of this ServerInterface when delayed creation is required Used by the code when delayed creation of the parameters is needed Users should not call this function

void com.vertica.sdk.ServerInterface.setSessionParamReader (ParamReader *sessionParamReader*)

Set the sessionParamReader of this ServerInterface

abstract void com.vertica.sdk.ServerInterface.vlog (String *logEntry*) `[protected]`, `[pure virtual]`

Callback for logging, implemented by the server

Referenced by com.vertica.sdk.ServerInterface.log().

Member Data Documentation

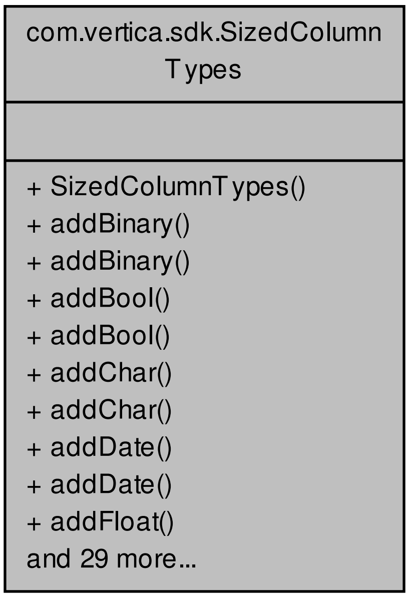
FileManager com.vertica.sdk.ServerInterface.fileManager

File manager of the session context

com.vertica.sdk.SizedColumnTypes Class Reference

Represents types and information to determine the size of the columns as input/output of a User Defined Function/-Transform.

Collaboration diagram for com.vertica.sdk.SizedColumnTypes:



Classes

- class **PartitionOrderColumnInfo**

  Represents the partition by and order by column information for each phase in a multi-phase transform function.

Public Member Functions

- void addBinary (int len, String fieldName)

  Adds a column of type BINARY.
- void **addBinary** (int len)
- void addBool (String fieldName)

  Adds a column of type BOOLEAN.
- void **addBool** ()
- void addChar (int len, String fieldName)

  Adds a column of type CHAR.
- void **addChar** (int len)
- void addDate (String fieldName)

  Adds a column of type DATE.
- void **addDate** ()
- void addFloat (String fieldName)

  Adds a column of type FLOAT.
- void **addFloat** ()
- void addInt (String fieldName)

  Adds a column of type INTEGER.
- void **addInt** ()
- void addLongVarbinary (int len, String fieldName)

  Adds a column of type LONGVARBINARY.
- void **addLongVarbinary** (int len)
- void addLongVarchar (int len, String fieldName)

  Adds a column of type LONGVARCHAR.
- void **addLongVarchar** (int len)
- void addNumeric (int precision, int scale, String fieldName)

  Adds a column of type NUMERIC.
- void **addNumeric** (int precision, int scale)
- void addTime (int precision, String fieldName)

  Adds a column of type TIME.
- void **addTime** (int precision)
- void addTimestamp (String fieldName)

  Adds a column of type TIMESTAMP.
- void **addTimestamp** ()
- void addTimeTz (int precision, String fieldName)

  Adds a column of type TIME WITH TIMEZONE.
- void **addTimeTz** (int precision)
- void addVarbinary (int len, String fieldName)

  Adds a column of type VARBINARY.
- void **addVarbinary** (int len)
- void addVarchar (int len, String fieldName)

  Adds a column of type VARCHAR.
- void **addVarchar** (int len)
- void getArgumentColumns (ArrayList< Integer > cols)

  Retrieves indexes of argument columns. Indexes in cols can be used in conjunction with other functions, e.g. getColumnType(size_t) and getColumnName(size_t).

- int getColumnCount ()

  Returns the number of columns.
- String getColumnName (int idx)

  Returns the name of the column at the specified index.
- VerticaType getColumnType (int idx)

  Returns the type of the column at the specified index.
- int getLastOrderColumnIdx ()

  Gets the last ORDER BY column index.
- int getLastPartitionColumnIdx ()

  Gets the last PARTITION BY column index.
- boolean isOrderByColumn (int idx)

  Indicates whether the column at the specified index is an ORDER BY column.
- boolean isPartitionByColumn (int idx)

  Indicates whether the column at the specified index is a PARTITION BY column.
- void setPartitionOrderColumnIdx (int partition_idx, int order_idx)

  Sets the PARTITION BY and ORDER BY column indexes.
- void setPartitionOrderColumnIdx (SizedColumnTypes other)

  Sets the PARTITION BY and ORDER BY column indexes from another SizedColumnTypes object.

Detailed Description

Represents types and information to determine the size of the columns as input/output of a User Defined Function/-Transform.

This class is used to exchange size and precision information between Vertica and the user defined function/transform function. Vertica provides the user code with size/precision information about the particular data types that it has been called with, and expects the user code to provide size/precision information about what it will return.

Member Function Documentation

void com.vertica.sdk.SizedColumnTypes.addBinary (int *len*, String *fieldName*)

Adds a column of type BINARY.

Parameters

len	The length of the binary string.
fieldName	The name for the output column.

void com.vertica.sdk.SizedColumnTypes.addBool (String *fieldName*)

Adds a column of type BOOLEAN.

Parameters

fieldName	The name for the output column.

void com.vertica.sdk.SizedColumnTypes.addChar (int *len*, String *fieldName*)

Adds a column of type CHAR.

Parameters

len	The length of the string.
fieldName	The name for the output column.

void com.vertica.sdk.SizedColumnTypes.addDate (String *fieldName*)

Adds a column of type DATE.

Parameters

fieldName	The name for the output column.

void com.vertica.sdk.SizedColumnTypes.addFloat (String *fieldName*)

Adds a column of type FLOAT.

Parameters

fieldName	The name for the output column.

void com.vertica.sdk.SizedColumnTypes.addInt (String *fieldName*)

Adds a column of type INTEGER.

Parameters

fieldName	The name for the output column.

void com.vertica.sdk.SizedColumnTypes.addLongVarbinary (int *len,* String *fieldName*)

Adds a column of type LONGVARBINARY.

Parameters

len	The length of the binary string.
fieldName	The name for the output column.

void com.vertica.sdk.SizedColumnTypes.addLongVarchar (int *len,* String *fieldName*)

Adds a column of type LONGVARCHAR.

Parameters

len	The length of the string.
fieldName	The name for the output column.

void com.vertica.sdk.SizedColumnTypes.addNumeric (int *precision,* int *scale,* String *fieldName*)

Adds a column of type NUMERIC.

Parameters

precision	The precision for the numeric value.
scale	The scale for the numeric value.
fieldName	The name for the output column.

void com.vertica.sdk.SizedColumnTypes.addTime (int *precision,* String *fieldName*)

Adds a column of type TIME.

Parameters

precision	The precision for the time.
fieldName	The name for the output column.

void com.vertica.sdk.SizedColumnTypes.addTimestamp (String *fieldName*)

Adds a column of type TIMESTAMP.

Parameters

fieldName	The name for the output column.

void com.vertica.sdk.SizedColumnTypes.addTimeTz (int *precision,* String *fieldName*)

Adds a column of type TIME WITH TIMEZONE.

Parameters

precision	The precision for the time.
fieldName	The name for the output column.

void com.vertica.sdk.SizedColumnTypes.addVarbinary (int *len,* String *fieldName*)

Adds a column of type VARBINARY.

Parameters

len	The length of the binary string.
fieldName	The name for the output column.

void com.vertica.sdk.SizedColumnTypes.addVarchar (int *len,* String *fieldName*)

Adds a column of type VARCHAR.

Parameters

len	The length of the string.
fieldName	The name for the output column.

void com.vertica.sdk.SizedColumnTypes.getArgumentColumns (ArrayList$<$ Integer $>$ *cols*)

Retrieves indexes of argument columns. Indexes in cols can be used in conjunction with other functions, e.g. getColumnType(size_t) and getColumnName(size_t).

Parameters

cols	A vector to store the retrieved column indexes.

String com.vertica.sdk.SizedColumnTypes.getColumnName (int *idx*)

Returns the name of the column at the specified index.

Parameters

idx	The index of the column

VerticaType com.vertica.sdk.SizedColumnTypes.getColumnType (int *idx*)

Returns the type of the column at the specified index.

Parameters

idx	The index of the column

Returns

a VerticaType object describing the column's data type.

Referenced by com.vertica.sdk.BlockReader.getVNumeric().

boolean com.vertica.sdk.SizedColumnTypes.isOrderByColumn (int *idx*)

Indicates whether the column at the specified index is an ORDER BY column.

Parameters

idx	The index of the column

boolean com.vertica.sdk.SizedColumnTypes.isPartitionByColumn (int *idx*)

Indicates whether the column at the specified index is a PARTITION BY column.

Parameters

idx	The index of the column

void com.vertica.sdk.SizedColumnTypes.setPartitionOrderColumnIdx (int *partition_idx,* int *order_idx*)

Sets the PARTITION BY and ORDER BY column indexes.

Parameters

partition_idx	Index of the last partition-by column
order_idx	Index of the last order-by column

void com.vertica.sdk.SizedColumnTypes.setPartitionOrderColumnIdx (SizedColumnTypes *other*)

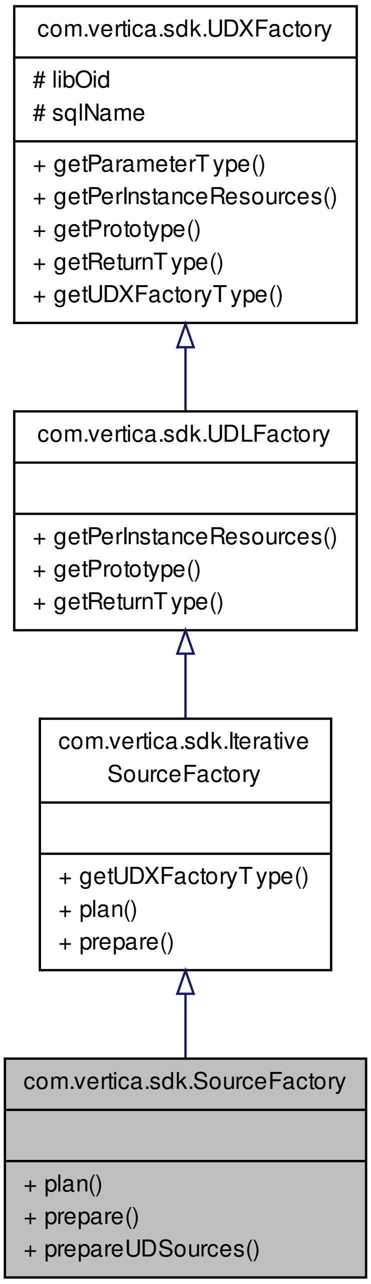
Sets the PARTITION BY and ORDER BY column indexes from another SizedColumnTypes object.

Parameters

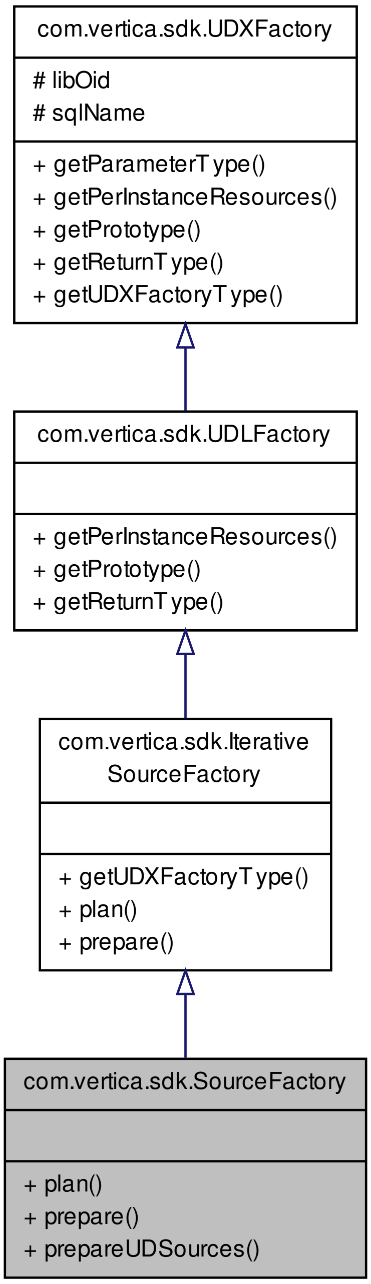
other	The SizedColumnTypes object to set the indexes from

com.vertica.sdk.SourceFactory Class Reference

Inheritance diagram for com.vertica.sdk.SourceFactory:



Collaboration diagram for com.vertica.sdk.SourceFactory:



Public Member Functions

- void getParameterType (ServerInterface srvInterface, SizedColumnTypes parameterTypes)
- void **getPerInstanceResources** (ServerInterface srvInterface, VResources res)
- void getPrototype (ServerInterface srvInterface, ColumnTypes argTypes, ColumnTypes returnType)
- void getReturnType (ServerInterface srvInterface, SizedColumnTypes argTypes, SizedColumnTypes returnType)
- UDXType getUDXFactoryType ()

- void plan (ServerInterface srvInterface, NodeSpecifyingPlanContext planCtxt) throws UdfException
- SourceIterator prepare (ServerInterface srvInterface, NodeSpecifyingPlanContext planCtxt) throws UdfException
- abstract ArrayList< UDSource > prepareUDSources (ServerInterface srvInterface, NodeSpecifyingPlanContext planCtxt) throws UdfException

Protected Attributes

- long **libOid**
- String **sqlName**

Detailed Description

A SourceFactory whose prepare() method constructs UDSources directly.

When implementing the factories for a UDSource, you have two options:

- Implement both an IterativeSourceFactory and a SourceIterator
- Implement just a SourceFactory (and no custom SourceIterator)

Member Function Documentation

void com.vertica.sdk.UDXFactory.getParameterType (ServerInterface *srvInterface,* SizedColumnTypes *parameterTypes*) [inherited]

Function to tell Vertica the name and types of parameters that this function uses. Vertica will use this to warn function callers that certain parameters they provide are not affecting anything, or that certain parameters that are not being set are reverting to default values.

void com.vertica.sdk.UDLFactory.getPrototype (ServerInterface *srvInterface,* ColumnTypes *argTypes,* ColumnTypes *returnType*) [virtual], [inherited]

Provides the argument and return types of the UDL. UDL's take no input tuples; as such, their prototype is empty.

Implements com.vertica.sdk.UDXFactory.

void com.vertica.sdk.UDLFactory.getReturnType (ServerInterface *srvInterface,* SizedColumnTypes *argTypes,* SizedColumnTypes *returnType*) [virtual], [inherited]

Not used in this form

Implements com.vertica.sdk.UDXFactory.

UDXType com.vertica.sdk.IterativeSourceFactory.getUDXFactoryType ( ) [virtual], [inherited]

Returns

the type of UDX Object instance this factory returns.

Note

User subclasses should use the appropriate subclass of UDXFactory and not override this method on their own.

Implements com.vertica.sdk.UDXFactory.

void com.vertica.sdk.SourceFactory.plan (ServerInterface *srvInterface,* NodeSpecifyingPlanContext *planCtxt*) throws UdfException

Execute any planning logic required at query plan time. This method is run once per query, during query initialization. Its job is to perform parameter validation, and to modify the set of nodes that the COPY statement will run on.

plan() runs exactly once per query, on the initiator node. If it throws an exception, the query will not proceed; it will be aborted prior to distributing the query to the other nodes and running prepare().

Parameters

srvInterface	Interface to server operations and functionality, including (not-per-column) parameter lookup
planCtxt	Context for storing and retrieving arbitrary data, for use just by this instance of this query. The same context is shared with plan(). Also provides functionality for specifying which nodes this query will run on.

Exceptions

UdfException	

SourceIterator com.vertica.sdk.SourceFactory.prepare (ServerInterface *srvInterface,* NodeSpecifyingPlanContext *planCtxt*) throws UdfException `[virtual]`

INTERNAL

Implements com.vertica.sdk.IterativeSourceFactory.

abstract ArrayList<UDSource> com.vertica.sdk.SourceFactory.prepareUDSources (ServerInterface *srvInterface,* NodeSpecifyingPlanContext *planCtxt*) throws UdfException `[pure virtual]`

Create UDSources. This function will be called on each node, prior to the Load operator starting to execute and prior to any other virtual functions on this class being called.

If necessary, it is safe for this method to store any of the argument references as local fields on this instance. All will persist for the duration of the query.

Unlike the standard SourceFactory, this method directly instantiates all of its UDSources, and returns a vector of them. This requires that all UDSources be resident in memory for the duration of the query, which is fine in the common case but which may not be acceptable for some resource-intensive UDSources.

Parameters

srvInterface	Interface to server operations and functionality, including (not-per-column) parameter lookup
planCtxt	Context for storing and retrieving arbitrary data, for use just by this instance of this query. The same context is shared with plan(). Also provides functionality for determining which nodes this query is running on.

Returns

A vector of UDSources to use for this query. Sources will be loaded in a pooled manner, several at a time.

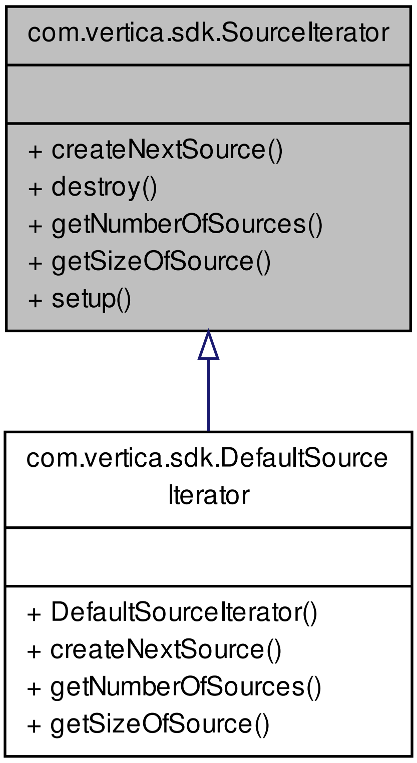
Exceptions

UdfException	

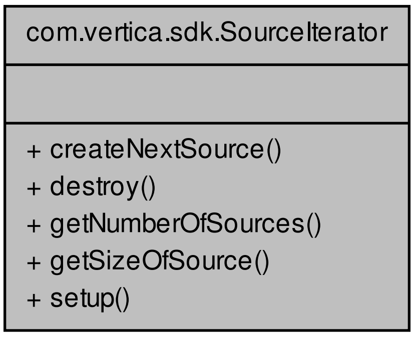
Referenced by com.vertica.sdk.SourceFactory.prepare().

com.vertica.sdk.SourceIterator Class Reference

Inheritance diagram for com.vertica.sdk.SourceIterator:



Collaboration diagram for com.vertica.sdk.SourceIterator:



Public Member Functions

- abstract UnsizedUDSource createNextSource (ServerInterface srvInterface) throws UdfException

    Create the next UDSource to process.
- void destroy (ServerInterface srvInterface, NodeSpecifyingPlanContext planCtxt) throws UdfException

    Tear down this SourceIterator.
- abstract int getNumberOfSources () throws UdfException
- Integer getSizeOfSource (int sourceNum) throws UdfException
- void setup (ServerInterface srvInterface, NodeSpecifyingPlanContext planCtxt) throws UdfException

    Set up this SourceIterator.

Detailed Description

Construct a set of Sources. createNextSource() will be called repeatedly until it returns NULL. Each resulting Source will be read to completion, and the contained data passed to the Filter and Parser.

Member Function Documentation

abstract UnsizedUDSource com.vertica.sdk.SourceIterator.createNextSource (ServerInterface *srvInterface*) throws UdfException `[pure virtual]`

Create the next UDSource to process.

Should return NULL if no further sources are available for processing.

Note that the previous Source may still be open and in use on a different thread when this function is called.

Returns

a new Source instance corresponding to a new input stream

Exceptions

UdfException	

Implemented in com.vertica.sdk.DefaultSourceIterator.

void com.vertica.sdk.SourceIterator.destroy (ServerInterface *srvInterface*, NodeSpecifyingPlanContext *planCtxt*) throws UdfException

Tear down this SourceIterator.

Should perform clean-up

Exceptions

UdfException	

abstract int com.vertica.sdk.SourceIterator.getNumberOfSources ( ) throws UdfException `[pure virtual]`

Returns

the total number of Sources that this factory will produce

Exceptions

UdfException	

Implemented in com.vertica.sdk.DefaultSourceIterator.

Integer com.vertica.sdk.SourceIterator.getSizeOfSource (int *sourceNum*) throws UdfException

Returns

the raw-data size of the sourceNum'th source that will be produced by createNextSource(). Should return `vint_null` if the size is unknown.

This value is used as a hint, and is used by the "load_streams" table to display load progress. If incorrect or not set, "load_streams" may contain incorrect or unhelpful information.

Exceptions

UdfException	

void com.vertica.sdk.SourceIterator.setup (ServerInterface *srvInterface*, NodeSpecifyingPlanContext *planCtxt*) throws UdfException

Set up this SourceIterator.

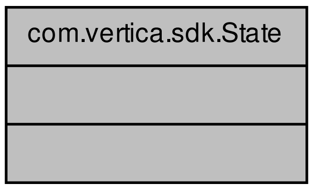
Should perform setup that should not take place in the constructor due to the exception-handling semantics of constructors

Exceptions

UdfException	

com.vertica.sdk.State Class Reference

Collaboration diagram for com.vertica.sdk.State:

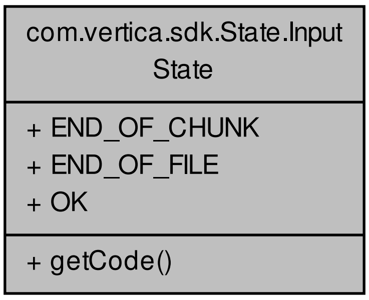


Classes

- enum InputState
- enum StreamState

com.vertica.sdk.State.InputState Enum Reference

Collaboration diagram for com.vertica.sdk.State.InputState:



Public Member Functions

- int **getCode** ()

Public Attributes

- **END_OF_CHUNK** =(2)
- **END_OF_FILE** =(1)
- **OK** =(0)

Detailed Description

InputState

Applies only to input streams; namely, UDFilter and UDParser.

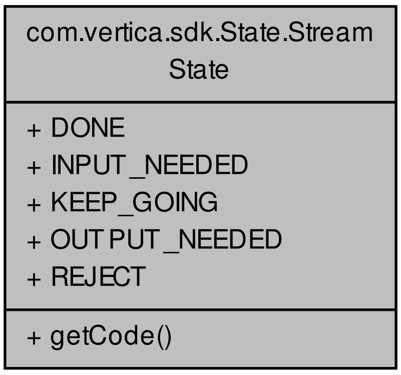
OK Currently at the start of or in the middle of a stream.

END_OF_FILE Reached the end of the input stream. No further data is available. Returning a StreamState of INPUT_NEEDED at this point is invalid, as there is no more input.

END_OF_CHUNK Reached the end of an input chunk. Applies only to a parser and only when fed by a Chunker, it means the current data block ends on a record boundary. In this state a parser should consume all data in the block before returning from process.

com.vertica.sdk.State.StreamState Enum Reference

Collaboration diagram for com.vertica.sdk.State.StreamState:



Public Member Functions

- int **getCode** ()

Public Attributes

- **DONE** =(2)
- **INPUT_NEEDED** =(0)
- **KEEP_GOING** =(4)
- **OUTPUT_NEEDED** =(1)
- **REJECT** =(3)

Detailed Description

StreamState

Indicates the state of a stream after process() has handled some input and some output data.

The different enum values have the following meanings:

INPUT_NEEDED Indicates that a stream is unable to continue without being given more input. It may not have consumed all of its available input yet. It does not need to have consumed every byte of input. Not valid for output-only streams, ie., UDSources.

OUTPUT_NEEDED Indicates that a stream is unable to write more output without being given a new or larger output buffer. Basically that it has "filled" the buffer as much as it is reasonably able to (which may be every byte full, some bytes full, even completely empty – in which case Vertica assumes that the UDL needs a larger buffer). Not valid for input-only streams, ie., UDParsers.

DONE The stream has completed; it will not be writing any more output nor consuming any more input.

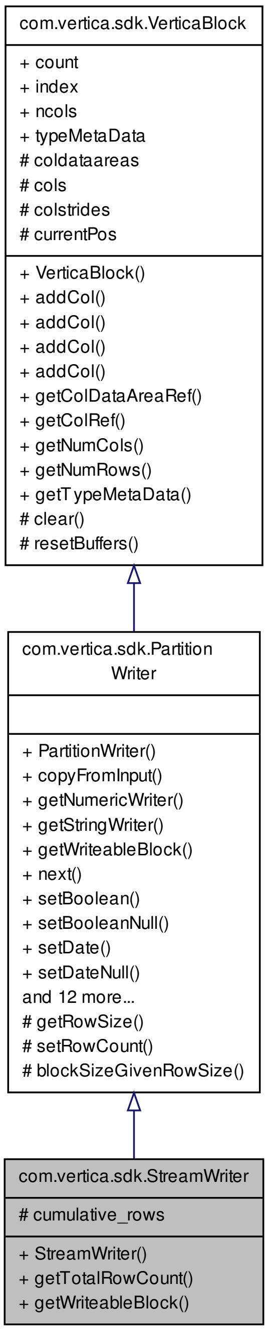
REJECT Only valid for UDParsers. Indicates that the Parser has consumed exactly one row, and that that row is invalid and should be processed as a rejected row.

KEEP_GOING Not commonly used. The stream has neither filled all of its output buffer nor consumed all of its input buffer, but would like to yield to the server. Typically it has neither consumed data nor produced data. This state should be used instead of a "wait" loop; a stream that is waiting for some external operation to complete should periodically return KEEP_GOING rather than simply blocking forever.

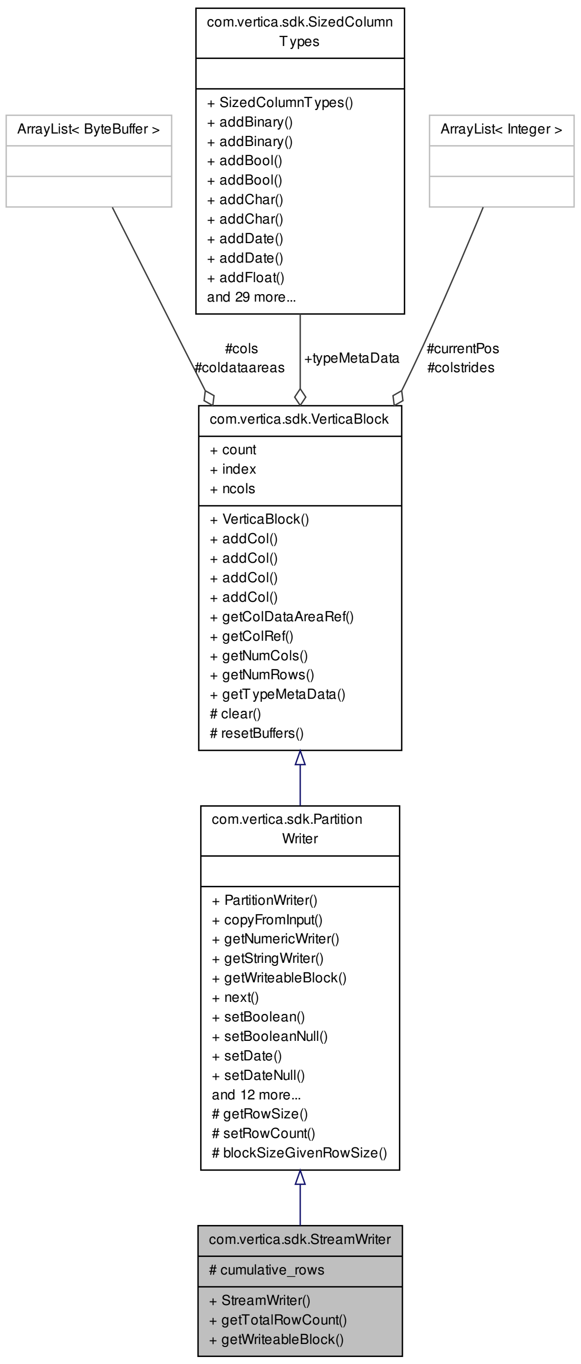
See the UDSource, UDFilter, and UDParser classes for how these streams are used.

com.vertica.sdk.StreamWriter Class Reference

Inheritance diagram for com.vertica.sdk.StreamWriter:



Collaboration diagram for com.vertica.sdk.StreamWriter:



Public Member Functions

- **StreamWriter** (int nargs)
- void addCol (ByteBuffer arg, int colstride, VerticaType dt, String colName)
- void addCol (ByteBuffer arg, int colstride, VerticaType dt)
- void addCol (ByteBuffer arg, ByteBuffer da, int colstride, VerticaType dt)
- void addCol (ByteBuffer arg, ByteBuffer da, int colstride, VerticaType dt, String colName)

- void copyFromInput (int dstIdx, PartitionReader input_reader, int srcIdx) throws UdfException
- ByteBuffer getColDataAreaRef (int idx)
- ByteBuffer getColRef (int idx)
- int getNumCols ()
- VNumeric **getNumericWriter** (int idx)
- int getNumRows ()
- VString **getStringWriter** (int idx)
- long **getTotalRowCount** ()
- SizedColumnTypes getTypeMetaData ()
- abstract boolean getWriteableBlock () throws UdfException, DestroyInvocation
- boolean **next** () throws UdfException, DestroyInvocation
- void **setBoolean** (int idx, boolean r)
- void **setBooleanNull** (int idx)
- void **setDate** (int idx, java.sql.Date r)
- void **setDateNull** (int idx)
- void **setDouble** (int idx, double r)
- void **setDoubleNull** (int idx)
- void setLong (int idx, long r)
- void **setLongNull** (int idx)
- void **setNumeric** (int idx, BigDecimal bd)
- void **setNumericNull** (int idx)
- void **setString** (int idx, String r)
- void **setStringNull** (int idx)
- void **setTimestamp** (int idx, java.sql.Timestamp r)
- void **setTimestampInfiniteNeg** (int idx)
- void **setTimestampInfinitePos** (int idx)
- void **setTimestampNull** (int idx)

Public Attributes

- int **count**
- int **index**
- int **ncols**
- SizedColumnTypes **typeMetaData**

Protected Member Functions

- void **clear** ()
- int **getRowSize** (SizedColumnTypes types)
- void **resetBuffers** ()
- void **setRowCount** (SizedColumnTypes types)

Static Protected Member Functions

- static int **blockSizeGivenRowSize** (int row_size)

Protected Attributes

- ArrayList< ByteBuffer > **coldataareas**
- ArrayList< ByteBuffer > **cols**
- ArrayList< Integer > **colstrides**
- long **cumulative_rows**
- ArrayList< Integer > **currentPos**

Detailed Description

StreamWriter provides an iterator-based write interface over output data for a stream of blocks. Automatically makes space a block-at-a-time, as needed.

Member Function Documentation

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, int *colstride*, VerticaType *dt*, String *colName*) `[inherited]`

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
colstride	The stride between data instances.
dt	The type of input.
colName	Name of the column

Referenced by com.vertica.sdk.VerticaBlock.addCol().

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, int *colstride*, VerticaType *dt*) `[inherited]`

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
colstride	The stride between data instances.
dt	The type of input.

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, ByteBuffer *da*, int *colstride*, VerticaType *dt*) `[inherited]`

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
da	The location to find out of band string data.
colstride	The stride between data instances.
dt	The type of input.

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, ByteBuffer *da*, int *colstride*, VerticaType *dt*, String *colName*) `[inherited]`

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
da	The location to find out of band string data.
colstride	The stride between data instances.

dt	The type of input.
colName	Name of the column

void com.vertica.sdk.PartitionWriter.copyFromInput (int *dstIdx*, PartitionReader *input_reader*, int *srcIdx*) throws UdfException `[inherited]`

Copies a column from the input reader to the output writer. The data types and sizes of the source and destination columns must match exactly.

Parameters

dstIdx	The destination column index (in the output writer)
input_reader	The input reader from which to copy a column
srcIdx	The source column index (in the input reader)

ByteBuffer com.vertica.sdk.VerticaBlock.getColDataAreaRef (int *idx*) `[inherited]`

Get the ByteBuffer that stores out of line string data (Data Area) for the idx'th argument

Parameters

idx	

Returns

Referenced by com.vertica.sdk.BlockReader.getVString().

ByteBuffer com.vertica.sdk.VerticaBlock.getColRef (int *idx*) `[inherited]`

Returns

a ByteBuffer to the idx'th argument, containing data for the column

Example:

```
* ByteBuffer a = arg_reader.getColPtr(0);
*
```

Referenced by com.vertica.sdk.PartitionWriter.copyFromInput(), com.vertica.sdk.BlockReader.getBoolean(), com.-vertica.sdk.BlockReader.getDouble(), com.vertica.sdk.BlockReader.getLong(), com.vertica.sdk.BlockReader.-getStringLength(), com.vertica.sdk.BlockReader.getStringLoc(), com.vertica.sdk.BlockReader.getVNumeric(), com.vertica.sdk.BlockReader.getVString(), com.vertica.sdk.BlockWriter.getVStringWriter(), com.vertica.sdk.-BlockReader.isBooleanNull(), com.vertica.sdk.ParamWriter.setBool(), com.vertica.sdk.BlockWriter.setBoolean(), com.vertica.sdk.BlockWriter.setBooleanNull(), com.vertica.sdk.ParamWriter.setDouble(), com.vertica.sdk.Block-Writer.setDouble(), com.vertica.sdk.BlockWriter.setDoubleNull(), com.vertica.sdk.PartitionWriter.setLong(), com.-vertica.sdk.BlockWriter.setLongNull(), com.vertica.sdk.ParamWriter.setLongString(), com.vertica.sdk.BlockWriter.-setNumeric(), com.vertica.sdk.ParamWriter.setNumeric(), com.vertica.sdk.BlockWriter.setString(), com.vertica.-sdk.ParamWriter.setString(), and com.vertica.sdk.BlockWriter.setStringNull().

int com.vertica.sdk.VerticaBlock.getNumCols ( ) `[inherited]`

Returns

the number of arguments held by this reader.

int com.vertica.sdk.VerticaBlock.getNumRows ( ) `[inherited]`

Returns

the number of rows held by this block.

SizedColumnTypes com.vertica.sdk.VerticaBlock.getTypeMetaData ( ) `[inherited]`

Returns

information about the types and numbers of arguments

Referenced by com.vertica.sdk.ParamReader.getType().

abstract boolean com.vertica.sdk.StreamWriter.getWriteableBlock ( ) throws UdfException, DestroyInvocation `[pure virtual]`

Gets a writeable block of data and positions cursor at the beginning.

Implements com.vertica.sdk.PartitionWriter.

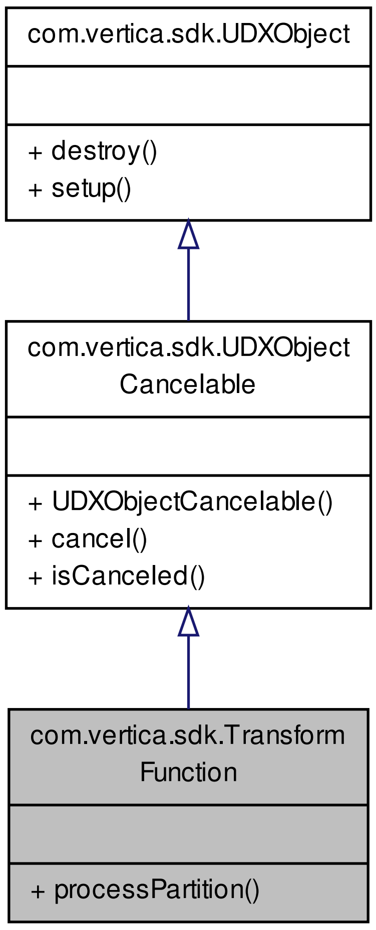
void com.vertica.sdk.PartitionWriter.setLong (int *idx*, long *r*) `[inherited]`

Setter methods

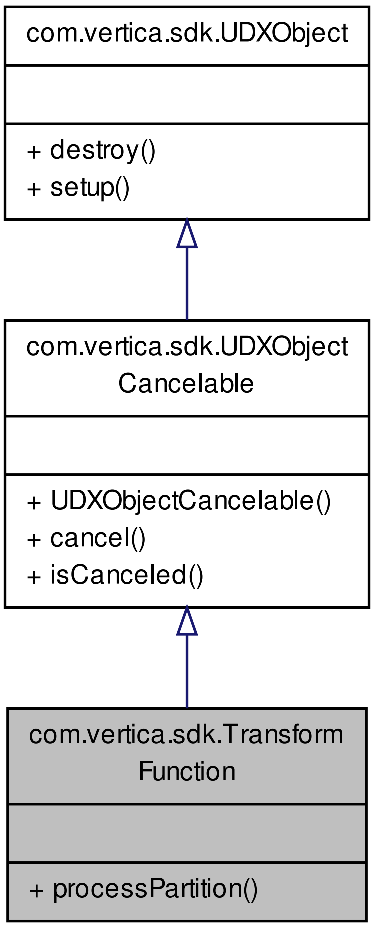
com.vertica.sdk.TransformFunction Class Reference

Interface for User Defined Transform, the actual code to process a partition of data coming in as a stream.

Inheritance diagram for com.vertica.sdk.TransformFunction:


com.vertica.sdk.UDXObject
+ destroy()
+ setup()
com.vertica.sdk.UDXObject Cancelable
+ UDXObjectCancelable()
+ cancel()
+ isCanceled()
com.vertica.sdk.Transform Function
+ processPartition()

Collaboration diagram for com.vertica.sdk.TransformFunction:


com.vertica.sdk.UDXObject
+ destroy()
+ setup()
com.vertica.sdk.UDXObject Cancelable
+ UDXObjectCancelable()
+ cancel()
+ isCanceled()
com.vertica.sdk.Transform Function
+ processPartition()

Public Member Functions

- void cancel (ServerInterface srvInterface)
- void destroy (ServerInterface srvInterface, SizedColumnTypes argTypes)
- boolean isCanceled ()
- abstract void processPartition (ServerInterface srvInterface, PartitionReader input_reader, PartitionWriter input_writer) throws UdfException, DestroyInvocation
- void setup (ServerInterface srvInterface, SizedColumnTypes argTypes)

Detailed Description

Interface for User Defined Transform, the actual code to process a partition of data coming in as a stream.

Member Function Documentation

void com.vertica.sdk.UDXObjectCancelable.cancel (ServerInterface *srvInterface*) `[inherited]`

This function is invoked from a different thread when the execution is canceled This baseclass cancel should be called in any override.

void com.vertica.sdk.UDXObject.destroy (ServerInterface *srvInterface,* SizedColumnTypes *argTypes*) `[inherited]`

Perform per instance destruction. This function may throw errors

boolean com.vertica.sdk.UDXObjectCancelable.isCanceled ( ) `[inherited]`

Returns true if execution was canceled.

abstract void com.vertica.sdk.TransformFunction.processPartition (ServerInterface *srvInterface,* PartitionReader *input_reader,* PartitionWriter *input_writer*) throws UdfException, DestroyInvocation `[pure virtual]`

Invoke a user defined transform on a set of rows. As the name suggests, a batch of rows are passed in for every invocation to amortize performance.

Parameters

srvInterface	a ServerInterface object used to communicate with Vertica
input_reader	input rows
output_writer	output location

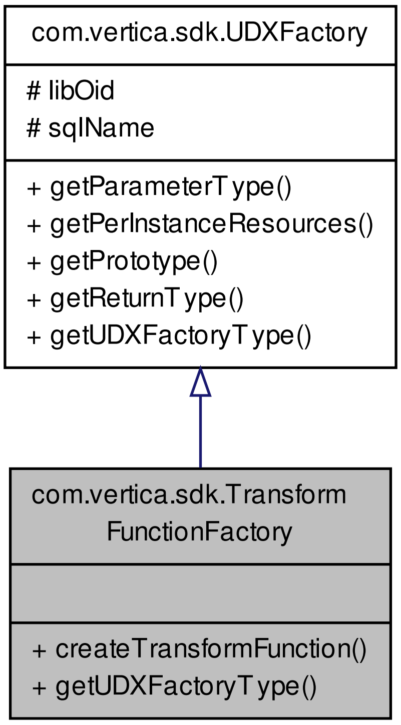
void com.vertica.sdk.UDXObject.setup (ServerInterface *srvInterface,* SizedColumnTypes *argTypes*) `[inherited]`

Perform per instance initialization. This function may throw errors.

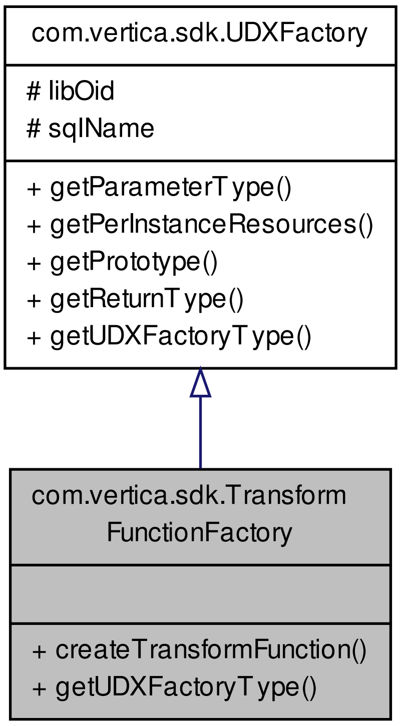
com.vertica.sdk.TransformFunctionFactory Class Reference

Interface to provide User Defined Transform compile time information.

Inheritance diagram for com.vertica.sdk.TransformFunctionFactory:


com.vertica.sdk.UDXFactory
# libOid
# sqlName
+ getParameterType()
+ getPerInstanceResources()
+ getPrototype()
+ getReturnType()
+ getUDXFactoryType()
com.vertica.sdk.TransformFunctionFactory
+ createTransformFunction()
+ getUDXFactoryType()

Collaboration diagram for com.vertica.sdk.TransformFunctionFactory:


com.vertica.sdk.UDXFactory
# libOid
# sqlName
+ getParameterType()
+ getPerInstanceResources()
+ getPrototype()
+ getReturnType()
+ getUDXFactoryType()
com.vertica.sdk.Transform FunctionFactory
+ createTransformFunction()
+ getUDXFactoryType()

Public Member Functions

- abstract TransformFunction createTransformFunction (ServerInterface srvInterface)
- void getParameterType (ServerInterface srvInterface, SizedColumnTypes parameterTypes)
- void getPerInstanceResources (ServerInterface srvInterface, VResources res)
- abstract void getPrototype (ServerInterface srvInterface, ColumnTypes argTypes, ColumnTypes returnType)
- abstract void getReturnType (ServerInterface srvInterface, SizedColumnTypes argTypes, SizedColumnTypes returnType) throws UdfException
- UDXType getUDXFactoryType ()

Protected Attributes

- long **libOid**
- String **sqlName**

Detailed Description

Interface to provide User Defined Transform compile time information.

Member Function Documentation

abstract TransformFunction com.vertica.sdk.TransformFunctionFactory.createTransformFunction (ServerInterface *srvInterface*) [pure virtual]

Called when Vertica needs a new TransformFunction object to process a UDTF function call.

Returns

an TransformFunction object which implements the UDx API described by this metadata.

Parameters

srvInterface	a ServerInterface object used to communicate with Vertica

Note

More than one object may be instantiated per query.

void com.vertica.sdk.UDXFactory.getParameterType (ServerInterface *srvInterface*, SizedColumnTypes *parameterTypes*) [inherited]

Function to tell Vertica the name and types of parameters that this function uses. Vertica will use this to warn function callers that certain parameters they provide are not affecting anything, or that certain parameters that are not being set are reverting to default values.

void com.vertica.sdk.UDXFactory.getPerInstanceResources (ServerInterface *srvInterface*, VResources *res*) [inherited]

Set the resource required for each instance of the UDX Object subclass

Parameters

srvInterface	a ServerInterface object used to communicate with Vertica
res	a VResources object used to tell Vertica what resources are needed by the UDX

abstract void com.vertica.sdk.UDXFactory.getPrototype (ServerInterface *srvInterface*, ColumnTypes *argTypes*, ColumnTypes *returnType*) [pure virtual], [inherited]

Provides the argument and return types of the UDX

Implemented in com.vertica.sdk.UDLFactory.

Referenced by com.vertica.sdk.ScalarFunctionFactory.getReturnType().

abstract void com.vertica.sdk.UDXFactory.getReturnType (ServerInterface *srvInterface*, SizedColumnTypes *argTypes*, SizedColumnTypes *returnType*) throws UdfException [pure virtual], [inherited]

Function to tell Vertica what the return types (and length/precision if necessary) of this UDX are.

For CHAR/VARCHAR types, specify the max length,

For NUMERIC types, specify the precision and scale.

For Time types (with or without time zone), specify the precision, -1 means unspecified/don't care

For IntervalYM/IntervalDS types, specify the precision and range

For all other types, no length/precision specification needed

Parameters

argTypes	Provides the data types of arguments that this UDT was called with. This may be used to modify the return types accordingly.
returnType	User code must fill in the names and data types returned by the UDT.

Implemented in com.vertica.sdk.ScalarFunctionFactory, and com.vertica.sdk.UDLFactory.

UDXType com.vertica.sdk.TransformFunctionFactory.getUDXFactoryType ( ) `[virtual]`

Returns

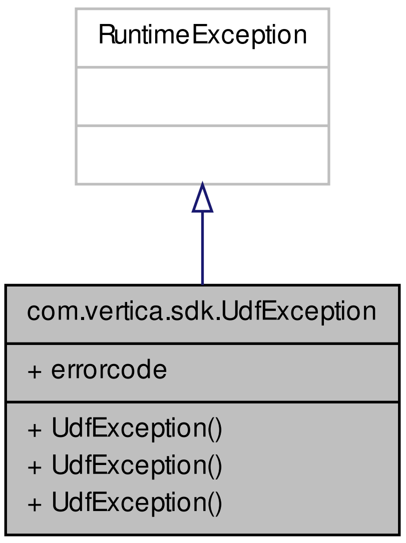
the object type internally used by Vertica

Implements com.vertica.sdk.UDXFactory.

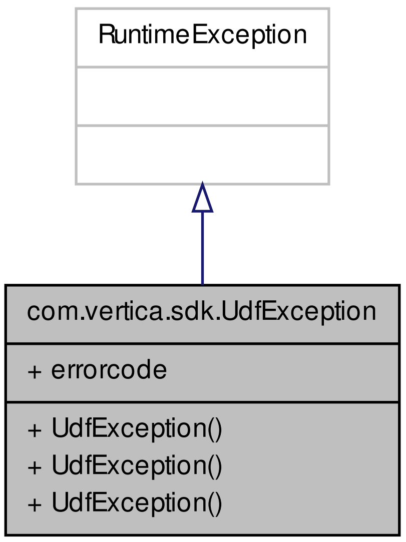
com.vertica.sdk.UdfException Class Reference

Contains error information, UDx code can throw object of this class to Vertica to indicate an error.

Inheritance diagram for com.vertica.sdk.UdfException:



Collaboration diagram for com.vertica.sdk.UdfException:



Public Member Functions

- UdfException (int errorcode, Throwable causedBy)
- UdfException (int errorcode, String message, Throwable causedBy)
- UdfException (int errorcode, String message)

Public Attributes

- int **errorcode**

Detailed Description

Contains error information, UDx code can throw object of this class to Vertica to indicate an error.

Constructor & Destructor Documentation

com.vertica.sdk.UdfException.UdfException (int *errorcode*, Throwable *causedBy*)

Constructor

Parameters

errorcode	a numeric id that UDx can use to indicate the error.
causedBy	an uncaught Throwable that caused the UDF to fail

com.vertica.sdk.UdfException.UdfException (int *errorcode,* String *message,* Throwable *causedBy*)

Constructor

Parameters

errorcode	a numeric id that UDx can use to indicate the error.
message	a human readable error message
causedBy	an uncaught Throwable that caused the UDF to fail

com.vertica.sdk.UdfException.UdfException (int *errorcode,* String *message*)

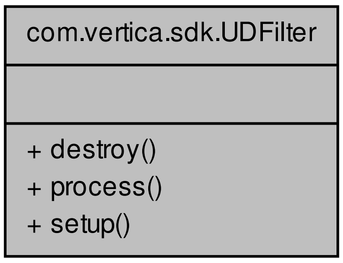
Constructor

Parameters

errorcode	a numeric id that UDx can use to indicate the error.
message	a human readable error message.

com.vertica.sdk.UDFilter Class Reference

Collaboration diagram for com.vertica.sdk.UDFilter:



Public Member Functions

- void destroy (ServerInterface srvInterface) throws UdfException
- abstract StreamState process (ServerInterface srvInterface, DataBuffer input, InputState input_state, DataBuffer output) throws UdfException
- void setup (ServerInterface srvInterface) throws UdfException

Detailed Description

UDFilter

Responsible for reading raw input data from a file and preparing it to be processed by a parser. This preparation may involve decompression, re-encoding, or any other sort of binary manipulation.

Member Function Documentation

void com.vertica.sdk.UDFilter.destroy (ServerInterface *srvInterface*) throws UdfException

UDFilter::destroy()

Will be invoked during query execution, after the last time that process() is called on this UDFilter instance for a particular input file.

May optionally be overridden to perform tear-down/destruction.

See UDFilter::setup() for a note about the restartability of UDFilters.

Exceptions

UdfException	

abstract StreamState com.vertica.sdk.UDFilter.process (ServerInterface *srvInterface,* DataBuffer *input,* InputState *input_state,* DataBuffer *output*) throws UdfException `[pure virtual]`

UDFilter::process()

Will be invoked repeatedly during query execution, until it returns DONE or until the query is canceled by the user.

On each invocation, process() is handed some input data and a buffer to write output data into. It is expected to read and process some amount of the input data, write some amount of output data, and return a value that informs Vertica what needs to happen next.

process() must set `input.offset` to the number of bytes that were successfully read from the `input` buffer, and that will not need to be re-consumed by a subsequent invocation of process(). This may not be larger than `input.size`. (`input.size` is the size of the buffer.) If it is set to 0, this indicates that process() cannot process any part of an input buffer of this size, and requires more data per invocation. (For example, a block-based decompression algorithm might return 0 if the input buffer does not contain a complete block.)

Note that `input` may contain null bytes, if the source file contains null bytes. Note also that `input` is NOT automatically null-terminated.

If `input_state` == END_OF_FILE, then the last byte in `input` is the last byte in the input stream. Returning INPUT_NEEDED will not result in any new input appearing. process() should return DONE in this case as soon as this operator has finished producing all output that it is going to produce.

process() must set `output.offset` to the number of bytes that were written to the `output` buffer. This may not be larger than `output.size`. If it is set to 0, this indicates that process() requires a larger output buffer.

Note that, unless OUTPUT_NEEDED is returned, `output` will be UNMODIFIED the next time process() is called. This means that pointers into the buffer will continue to be valid. It also means that `output.offset` may be set. So, in general, process() code should assume that buffers start at `output.buf[output.offset]`. The same goes for `input` and INPUT_NEEDED. Note also that, as a performance optimization, upstream operators may start processing emitted data (data between output.buf[0] and output.buf[output.offset]) before OUTPUT_NEEDED is returned. For this reason, `output.offset` must be strictly increasing.

process() must not block indefinitely. If it cannot proceed for an extended period of time, it should return KEEP_GOING. It will be called again shortly. Failure to do this will, among other things, prevent the query from being canceled by the user.

Returns

OUTPUT_NEEDED if this UDFilter has more data to produce; INPUT_NEEDED if it needs more data to continue working; DONE if has no more data to produce.

Note that it is UNSAFE to maintain pointers or references to any of these arguments (or any other argument passed by reference into any other function in this API) beyond the scope of the function call in question. For example, do not store a reference to the server interface or the input block on an instance variable. Vertica may free and replace these objects.

Exceptions

UdfException	

void com.vertica.sdk.UDFilter.setup (ServerInterface *srvInterface*) throws UdfException

UDFilter::setup()

Will be invoked during query execution, prior to the first time that process() is called on this UDFilter instance for a particular input file.

May optionally be overridden to perform setup/initialzation.

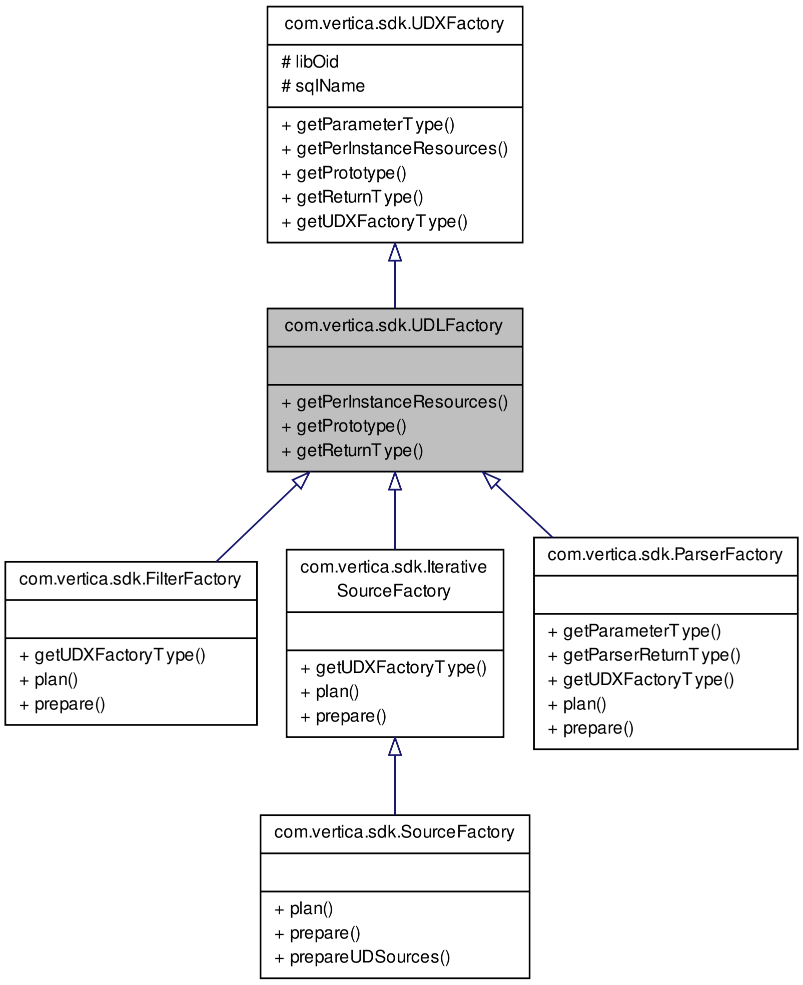
Note that UDFilters MUST BE RESTARTABLE! If loading large numbers of files, a given UDFilter may be re-used for multiple files. Vertica follows the worker-pool design pattern: At the start of COPY execution, several Parsers and several Filters are instantiated per node, by calling the corresponding prepare() method multiple times. Each Filter/Parser pair is then internally assigned to an initial Source (UDSource or internal). At that point, setup() is called; then process() is called until it is finished; then destroy() is called. If there are still sources in the pool waiting to be processed, then the UDFilter/UDSource pair will be given a second Source; setup() will be called a second time, then process() until it is finished, then destroy(). This repeats until all sources have been read.

Exceptions

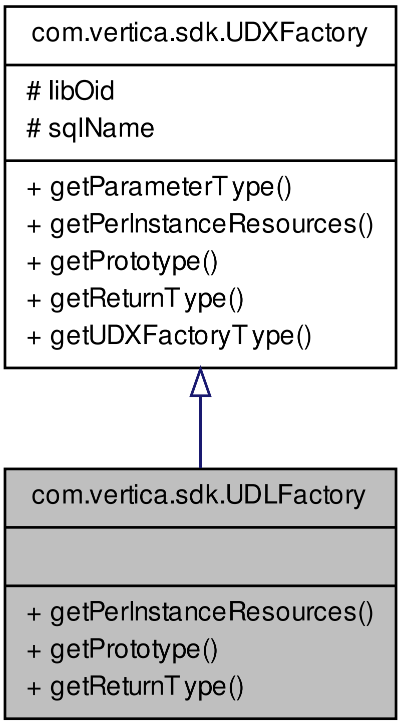
UdfException	

com.vertica.sdk.UDLFactory Class Reference

Inheritance diagram for com.vertica.sdk.UDLFactory:



Collaboration diagram for com.vertica.sdk.UDLFactory:



Public Member Functions

- void getParameterType (ServerInterface srvInterface, SizedColumnTypes parameterTypes)
- void **getPerInstanceResources** (ServerInterface srvInterface, VResources res)
- void getPrototype (ServerInterface srvInterface, ColumnTypes argTypes, ColumnTypes returnType)
- void getReturnType (ServerInterface srvInterface, SizedColumnTypes argTypes, SizedColumnTypes return-Type)
- abstract UDXType getUDXFactoryType ()

Protected Attributes

- long **libOid**
- String **sqlName**

Member Function Documentation

void com.vertica.sdk.UDXFactory.getParameterType (ServerInterface *srvInterface,* SizedColumnTypes *parameterTypes*) `[inherited]`

Function to tell Vertica the name and types of parameters that this function uses. Vertica will use this to warn function callers that certain parameters they provide are not affecting anything, or that certain parameters that are not being set are reverting to default values.

void com.vertica.sdk.UDLFactory.getPrototype (ServerInterface *srvInterface*, ColumnTypes *argTypes*, ColumnTypes *returnType*) `[virtual]`

Provides the argument and return types of the UDL. UDL's take no input tuples; as such, their prototype is empty.

Implements com.vertica.sdk.UDXFactory.

void com.vertica.sdk.UDLFactory.getReturnType (ServerInterface *srvInterface*, SizedColumnTypes *argTypes*, SizedColumnTypes *returnType*) `[virtual]`

Not used in this form

Implements com.vertica.sdk.UDXFactory.

abstract UDXType com.vertica.sdk.UDXFactory.getUDXFactoryType ( ) `[pure virtual], [inherited]`

Returns

the type of UDX Object instance this factory returns.

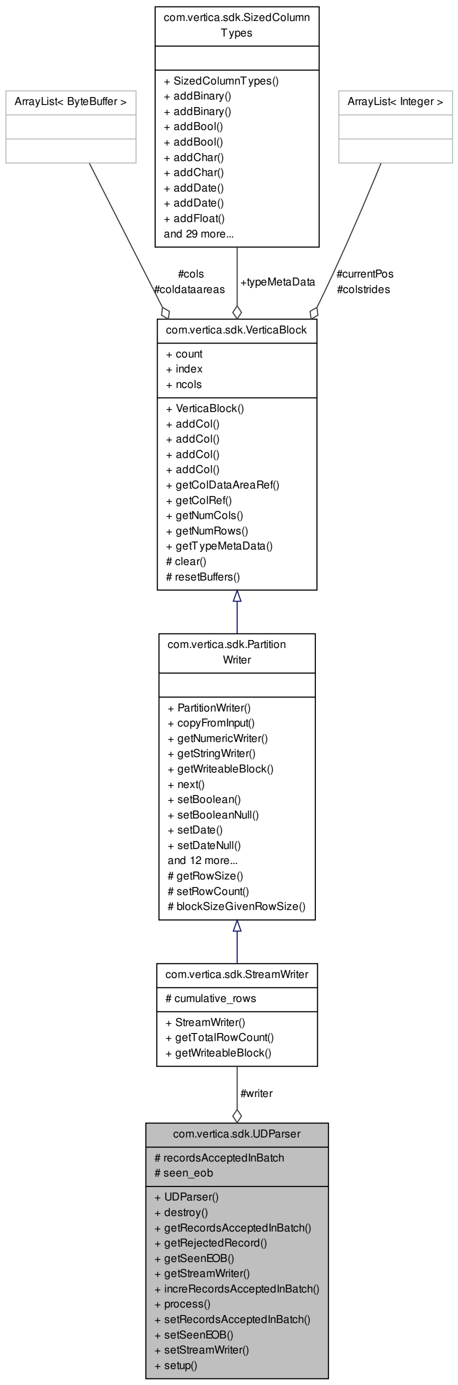
Note

User subclasses should use the appropriate subclass of UDXFactory and not override this method on their own.

Implemented in com.vertica.sdk.ScalarFunctionFactory, com.vertica.sdk.TransformFunctionFactory, com.vertica.sdk.IterativeSourceFactory, com.vertica.sdk.FilterFactory, and com.vertica.sdk.ParserFactory.

com.vertica.sdk.UDParser Class Reference

Collaboration diagram for com.vertica.sdk.UDParser:



Public Member Functions

- void destroy (ServerInterface srvInterface, SizedColumnTypes returnType) throws UdfException
- int **getRecordsAcceptedInBatch** ()
- RejectedRecord getRejectedRecord () throws UdfException

- boolean **getSeenEOB** ()
- StreamWriter **getStreamWriter** ()
- void **increRecordsAcceptedInBatch** ()
- abstract StreamState process (ServerInterface srvInterface, DataBuffer input, InputState input_state) throws UdfException, DestroyInvocation
- void **setRecordsAcceptedInBatch** (int i)
- void **setSeenEOB** (Boolean b)
- void **setStreamWriter** (StreamWriter writer)
- void setup (ServerInterface srvInterface, SizedColumnTypes returnType) throws UdfException

Protected Attributes

- int **recordsAcceptedInBatch**
- boolean **seen_eob**
- StreamWriter writer

Detailed Description

UDParser

Responsible for parsing an input stream into Vertica tuples, rows to be inserted into a table.

Member Function Documentation

void com.vertica.sdk.UDParser.destroy (**ServerInterface** *srvInterface,* **SizedColumnTypes** *returnType*) throws **UdfException**

UDParser::destroy()

Will be invoked during query execution, after the last time that process() is called on this UDParser instance for a particular input file.

May optionally be overridden to perform tear-down/destruction.

See UDParser::setup() for a note about the restartability of UDParsers.

Exceptions

UdfException	

RejectedRecord com.vertica.sdk.UDParser.getRejectedRecord ( ) throws **UdfException**

Returns information about the rejected record

Exceptions

UdfException	

abstract **StreamState** com.vertica.sdk.UDParser.process (**ServerInterface** *srvInterface,* **DataBuffer** *input,* **InputState** *input_state*) throws **UdfException, DestroyInvocation** `[pure virtual]`

UDParser::prepareToCooperate()

Notification to this parser that it should prepare to share parsing input with another. This can only happen when a parser has an associated chunker. Default implementation does nothing. UDParser::isReadyToCooperate()

Called after UDParser::prepareToCooperate(), returns false if this parser is not yet ready to cooperate. Once this method returns true the parser can begin to cooperate. Default implementation returns true, override if some preparation is required before the parser can cooperate (e.g. a certain # of rows must be skipped). UDParser::process()

Will be invoked repeatedly during query execution, until it returns DONE or until the query is canceled by the user.

On each invocation, process() will be given an input buffer. It should read data from that buffer, converting it to fields and tuples and writing those tuples via `writer`. Once it has consumed as much as it reasonably can (for example, once it has consumed the last complete row in the input buffer), it should return INPUT_NEEDED to indicate that more data is needed, or DONE to indicate that it has completed parsing this input stream and will not be reading more bytes from it.

If `input_state` == END_OF_FILE, then the last byte in `input` is the last byte in the input stream. Returning INPUT_NEEDED will not result in any new input appearing. process() should return DONE in this case as soon as this operator has finished producing all output that it is going to produce.

Note that `input` may contain null bytes, if the source file contains null bytes. Note also that `input` is NOT automatically null-terminated.

process() must not block indefinitely. If it cannot proceed for an extended period of time, it should return KEEP_GOING. It will be called again shortly. Failure to do this will, among other things, prevent the query from being canceled by the user.

Note that, unless INPUT_NEEDED is returned, `input` will be UNMODIFIED the next time process() is called. This means that pointers into the buffer will continue to be valid. It also means that `input.offset` may be set. So, in general, process() code should assume that buffers start at `input.buf[input.offset]`.

Row Rejection

process() can "reject" a row, causing it to be logged by Vertica's rejected-rows mechanism. Rejected rows should not be emitted as tuples. A rejected row must start at the first byte of `input` (meaning all previous input must have been consumed by a previous call to process()). To reject a row, set `input.offset` to the size of the row, and return REJECT.

Returns

INPUT_NEEDED if this UDParser has more data to produce; DONE if has no more data to produce; REJECT to reject a row

Note that it is UNSAFE to maintain pointers or references to any of these arguments (or any other argument passed by reference into any other function in this API) beyond the scope of the function call in question. For example, do not store a reference to the server interface or the input block on an instance variable. Vertica may free and replace these objects.

Exceptions

UdfException	

void com.vertica.sdk.UDParser.setup (ServerInterface *srvInterface,* SizedColumnTypes *returnType*) throws UdfException

UDParser::setup()

Will be invoked during query execution, prior to the first time that process() is called on this UDParser instance for a particular input source.

May optionally be overridden to perform setup/initialization.

Note that UDParsers MUST BE RESTARTABLE! If loading large numbers of files, a given UDParsers may be re-used for multiple files. Vertica follows the worker-pool design pattern: At the start of COPY execution, several Parsers and several Filters are instantiated per node, by calling the corresponding prepare() method multiple times. Each Filter/Parser pair is then internally assigned to an initial Source (UDSource or internal). At that point, setup() is called; then process() is called until it is finished; then destroy() is called. If there are still sources in the pool waiting to be processed, then the UDFilter/UDSource pair will be given a second Source; setup() will be called a second time, then process() until it is finished, then destroy(). This repeats until all sources have been read.

Exceptions

UdfException	

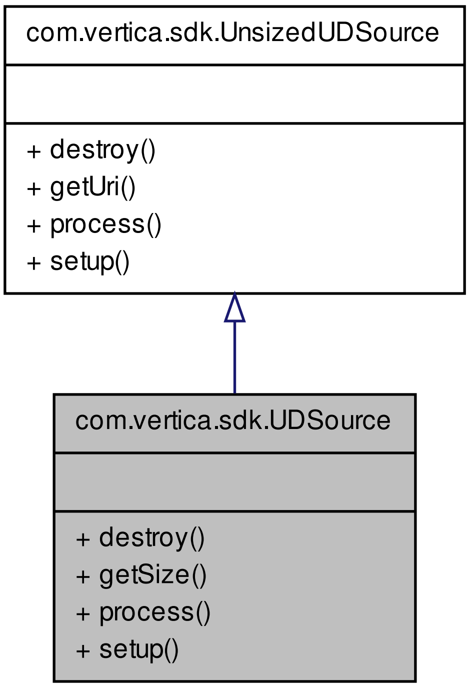
Member Data Documentation

StreamWriter com.vertica.sdk.UDParser.writer `[protected]`

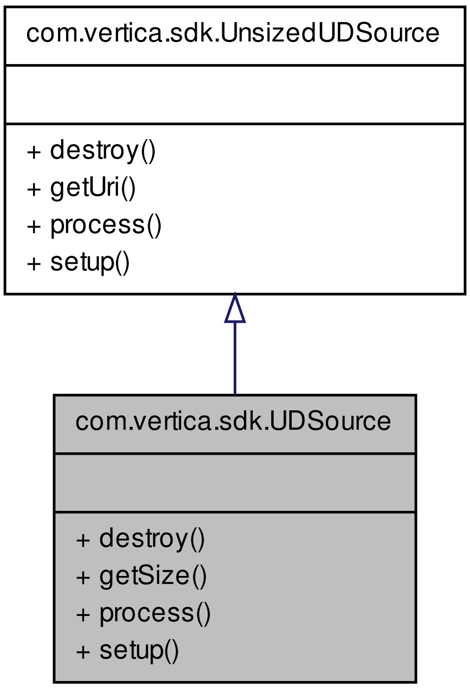
Writer to write parsed tuples to. Has the same API as PartitionWriter, from the UDT framework.

com.vertica.sdk.UDSource Class Reference

Inheritance diagram for com.vertica.sdk.UDSource:



Collaboration diagram for com.vertica.sdk.UDSource:



Public Member Functions

- void destroy (ServerInterface srvInterface) throws UdfException
- Integer getSize ()
- String getUri ()
- abstract StreamState process (ServerInterface srvInterface, DataBuffer output) throws UdfException
- void setup (ServerInterface srvInterface) throws UdfException

Detailed Description

UDSource

Responsible for acquiring data from an external source (such as a file, a URL, etc) and producing that data in a streaming manner.

Member Function Documentation

void com.vertica.sdk.UDSource.destroy (ServerInterface *srvInterface*) throws UdfException

UDSource::destroy()

Will be invoked during query execution, after the last time that process() is called on this UDSource instance.

May optionally be overridden to perform tear-down/destruction.

Exceptions

UdfException	

Integer com.vertica.sdk.UDSource.getSize ( )

UDSource::getSize()

Returns the estimated number of bytes that process() will return.

This value is treated as advisory only. It is used to indicate the file size in the LOAD_STREAMS table.

IMPORTANT: getSize() can be called at any time, even before setup() is called! (Though not before or during the constructor.)

In the case of Sources whose factories can potentially produce many UDSource instances, getSize() should avoid acquiring resources that last for the life of the object. Doing otherwise can defeat Vertica's attempts to limit the maximum number of Sources that are consuming system resources at any given time. For example, if it opens a file handle and leaves that file handle open for use by process(), and if a large number of UDSources are loaded in a single statement, the query may exceed the operating system limit on file handles and crash, even though Vertica only operates on a small number of files at once. This doesn't apply to singleton Sources, Sources whose factory will only ever produce one UDSource instance.

String com.vertica.sdk.UnsizedUDSource.getUri ( ) `[inherited]`

UnsizedUDSource::getUri()

Return the URI of the current source of data.

This function will be invoked during execution to fill in monitoring information.

abstract StreamState com.vertica.sdk.UDSource.process (ServerInterface *srvInterface*, DataBuffer *output*) throws UdfException `[pure virtual]`

UDSource::process()

Will be invoked repeatedly during query execution, until it returns DONE or until the query is canceled by the user.

On each invocation, process() should acquire more data and write that data to the buffer specified by `output`.

process() must set `output.offset` to the number of bytes that were written to the `output` buffer. It is common, though not necessary, for this to be the same as `output.size`. `output.offset` is initially uninitialized. If it is set to 0, this indicates that the `output` buffer is too small for process() to be able to write a unit of input to it.

Note that, unless OUTPUT_NEEDED is returned, `output` will be UNMODIFIED the next time process() is called. This means that pointers into the buffer will continue to be valid. It also means that `output.offset` may be set. So, in general, process() code should assume that buffers start at `output.buf[output.offset]`. Note also that, as a performance optimization, upstream operators may start processing emitted data (data between output.buf[0] and output.buf[output.offset]) before OUTPUT_NEEDED is returned. For this reason, `output.offset` must be strictly increasing.

process() must not block indefinitely. If it cannot proceed for an extended period of time, it should return KEEP_GOING. It will be called again shortly. Failure to do this will, among other things, prevent the query from being canceled by the user.

Returns

OUTPUT_NEEDED if this UDSource has more data to produce; DONE if has no more data to produce.

Note that it is UNSAFE to maintain pointers or references to any of these arguments (or any other argument passed by reference into any other function in this API) beyond the scope of the function call in question. For example, do not store a reference to the server interface or the input block on an instance variable. Vertica may free and replace these objects.

Exceptions

UdfException	

Implements com.vertica.sdk.UnsizedUDSource.

void com.vertica.sdk.UDSource.setup (ServerInterface *srvInterface*) throws UdfException

UDSource::setup()

Will be invoked during query execution, prior to the first time that process() is called on this UDSource instance.

May optionally be overridden to perform setup/initialization.

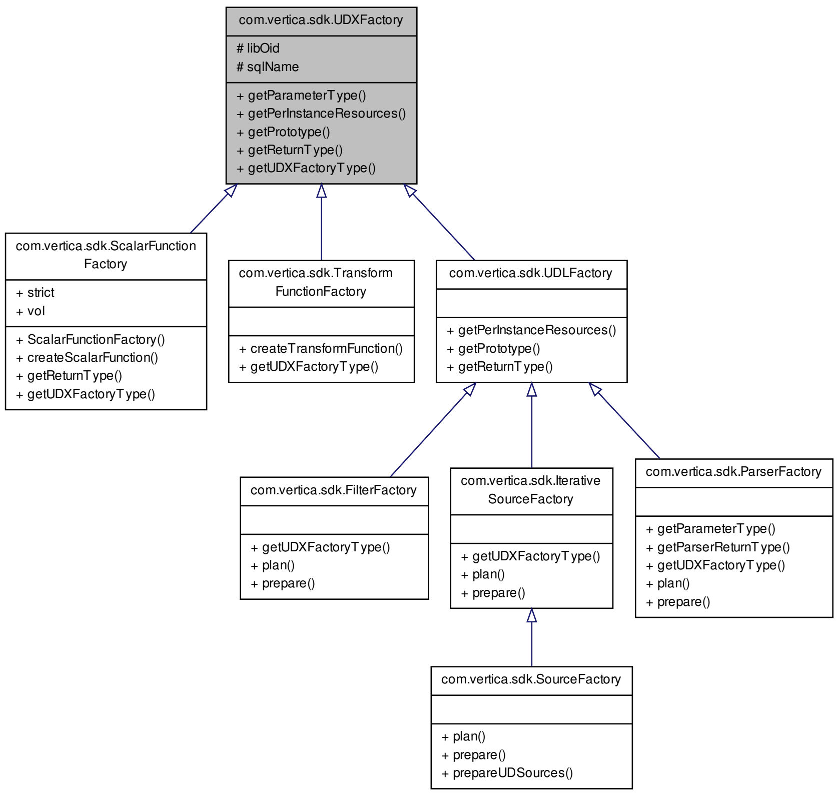
Exceptions

UdfException	

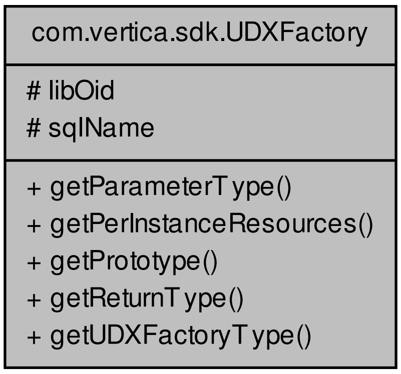
com.vertica.sdk.UDXFactory Class Reference

MetaData interface for Vertica User Defined extensions.

Inheritance diagram for com.vertica.sdk.UDXFactory:



Collaboration diagram for com.vertica.sdk.UDXFactory:



Classes

- enum UDXType

Public Member Functions

- void getParameterType (ServerInterface srvInterface, SizedColumnTypes parameterTypes)
- void getPerInstanceResources (ServerInterface srvInterface, VResources res)
- abstract void getPrototype (ServerInterface srvInterface, ColumnTypes argTypes, ColumnTypes returnType)
- abstract void getReturnType (ServerInterface srvInterface, SizedColumnTypes argTypes, SizedColumnTypes returnType) throws UdfException
- abstract UDXType getUDXFactoryType ()

Protected Attributes

- long **libOid**
- String **sqlName**

Detailed Description

MetaData interface for Vertica User Defined extensions.

Member Function Documentation

void com.vertica.sdk.UDXFactory.getParameterType (ServerInterface *srvInterface*, SizedColumnTypes *parameterTypes*)

Function to tell Vertica the name and types of parameters that this function uses. Vertica will use this to warn function callers that certain parameters they provide are not affecting anything, or that certain parameters that are not being set are reverting to default values.

void com.vertica.sdk.UDXFactory.getPerInstanceResources (ServerInterface *srvInterface,* VResources *res*)

Set the resource required for each instance of the UDX Object subclass

Parameters

srvInterface	a ServerInterface object used to communicate with Vertica
res	a VResources object used to tell Vertica what resources are needed by the UDX

abstract void com.vertica.sdk.UDXFactory.getPrototype (ServerInterface *srvInterface*, ColumnTypes *argTypes*, ColumnTypes *returnType*) `[pure virtual]`

Provides the argument and return types of the UDX

Implemented in com.vertica.sdk.UDLFactory.

Referenced by com.vertica.sdk.ScalarFunctionFactory.getReturnType().

abstract void com.vertica.sdk.UDXFactory.getReturnType (ServerInterface *srvInterface*, SizedColumnTypes *argTypes*, SizedColumnTypes *returnType*) throws UdfException `[pure virtual]`

Function to tell Vertica what the return types (and length/precision if necessary) of this UDX are.

For CHAR/VARCHAR types, specify the max length,

For NUMERIC types, specify the precision and scale.

For Time types (with or without time zone), specify the precision, -1 means unspecified/don't care

For IntervalYM/IntervalDS types, specify the precision and range

For all other types, no length/precision specification needed

Parameters

argTypes	Provides the data types of arguments that this UDT was called with. This may be used to modify the return types accordingly.
returnType	User code must fill in the names and data types returned by the UDT.

Implemented in com.vertica.sdk.ScalarFunctionFactory, and com.vertica.sdk.UDLFactory.

abstract UDXType com.vertica.sdk.UDXFactory.getUDXFactoryType ( ) `[pure virtual]`

Returns

the type of UDX Object instance this factory returns.

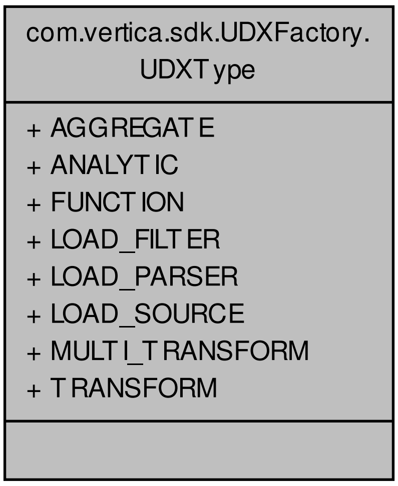
Note

User subclasses should use the appropriate subclass of UDXFactory and not override this method on their own.

Implemented in com.vertica.sdk.ScalarFunctionFactory, com.vertica.sdk.TransformFunctionFactory, com.vertica.-sdk.IterativeSourceFactory, com.vertica.sdk.FilterFactory, and com.vertica.sdk.ParserFactory.

com.vertica.sdk.UDXFactory.UDXType Enum Reference

Collaboration diagram for com.vertica.sdk.UDXFactory.UDXType:



Public Attributes

- **AGGREGATE**
- **ANALYTIC**
- **FUNCTION**
- **LOAD_FILTER**
- **LOAD_PARSER**
- **LOAD_SOURCE**
- **MULTI_TRANSFORM**
- **TRANSFORM**

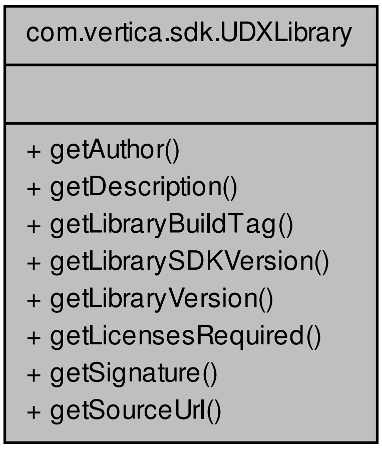
Detailed Description

The type of UDX instance this factory produces

com.vertica.sdk.UDXLibrary Class Reference

MetaData interface for Vertica User Defined extension libraries.

Collaboration diagram for com.vertica.sdk.UDXLibrary:



Public Member Functions

- String **getAuthor** ()
- String **getDescription** ()
- String **getLibraryBuildTag** ()
- String **getLibrarySDKVersion** ()
- String **getLibraryVersion** ()
- String **getLicensesRequired** ()
- String **getSignature** ()
- String **getSourceUrl** ()

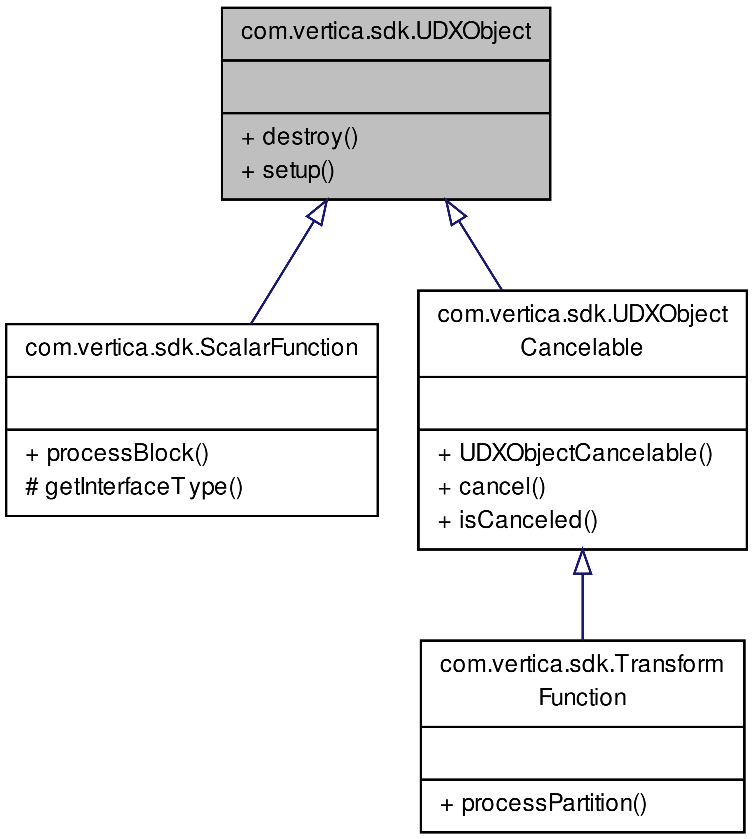
Detailed Description

MetaData interface for Vertica User Defined extension libraries.

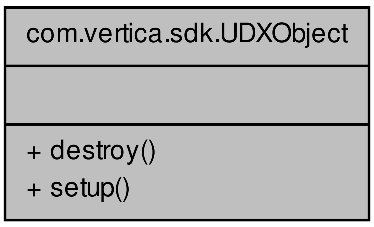
com.vertica.sdk.UDXObject Class Reference

Base class for Vertica User Defined extensions, the object themselves.

Inheritance diagram for com.vertica.sdk.UDXObject:


com.vertica.sdk.UDXObject
+ destroy()
+ setup()
com.vertica.sdk.ScalarFunction
+ processBlock()
# getInterfaceType()
com.vertica.sdk.UDXObject Cancelable
+ UDXObjectCancelable()
+ cancel()
+ isCanceled()
com.vertica.sdk.Transform Function
+ processPartition()

Collaboration diagram for com.vertica.sdk.UDXObject:



Public Member Functions

- void destroy (ServerInterface srvInterface, SizedColumnTypes argTypes)
- void setup (ServerInterface srvInterface, SizedColumnTypes argTypes)

Detailed Description

Base class for Vertica User Defined extensions, the object themselves.

Member Function Documentation

void com.vertica.sdk.UDXObject.destroy (ServerInterface *srvInterface,* SizedColumnTypes *argTypes*)

Perform per instance destruction. This function may throw errors

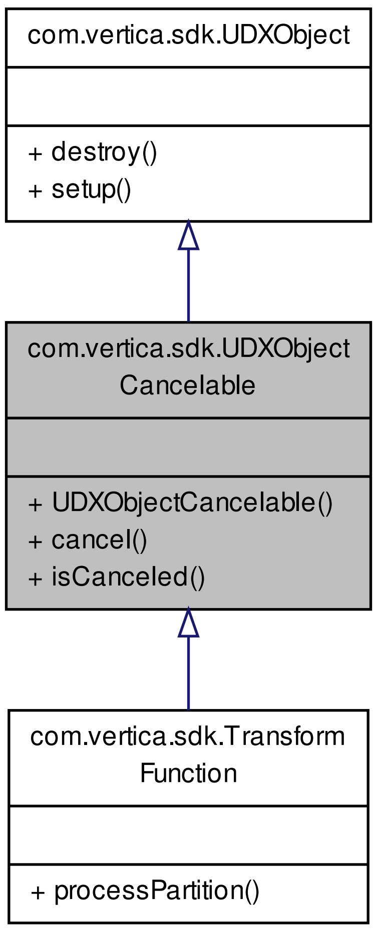
void com.vertica.sdk.UDXObject.setup (ServerInterface *srvInterface,* SizedColumnTypes *argTypes*)

Perform per instance initialization. This function may throw errors.

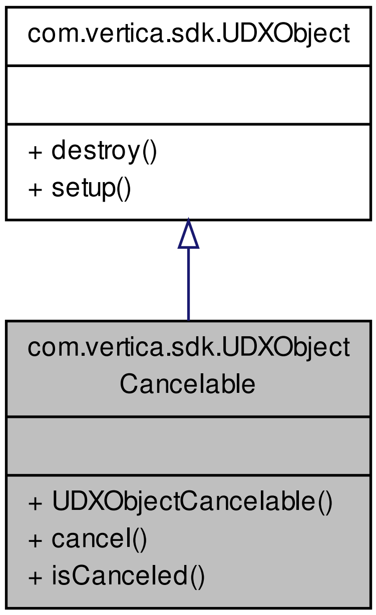
com.vertica.sdk.UDXObjectCancelable Class Reference

Base class for CANCELABLE Vertica User Defined extensions.

Inheritance diagram for com.vertica.sdk.UDXObjectCancelable:


com.vertica.sdk.UDXObject
+ destroy()
+ setup()
com.vertica.sdk.UDXObject Cancelable
+ UDXObjectCancelable()
+ cancel()
+ isCanceled()
com.vertica.sdk.Transform Function
+ processPartition()

Collaboration diagram for com.vertica.sdk.UDXObjectCancelable:



Public Member Functions

- void cancel (ServerInterface srvInterface)
- void destroy (ServerInterface srvInterface, SizedColumnTypes argTypes)
- boolean isCanceled ()
- void setup (ServerInterface srvInterface, SizedColumnTypes argTypes)

Detailed Description

Base class for CANCELABLE Vertica User Defined extensions.

Member Function Documentation

void com.vertica.sdk.UDXObjectCancelable.cancel (**ServerInterface** *srvInterface*)

This function is invoked from a different thread when the execution is canceled This baseclass cancel should be called in any override.

void com.vertica.sdk.UDXObject.destroy (**ServerInterface** *srvInterface,* **SizedColumnTypes** *argTypes*) `[inherited]`

Perform per instance destruction. This function may throw errors

boolean com.vertica.sdk.UDXObjectCancelable.isCanceled ( )

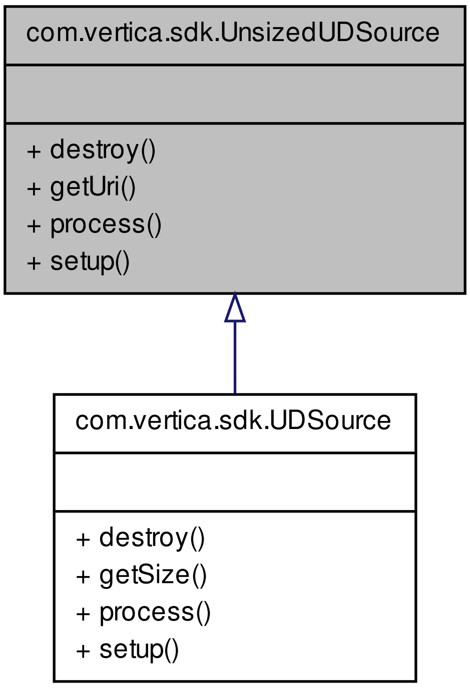
Returns true if execution was canceled.

void com.vertica.sdk.UDXObject.setup (ServerInterface *srvInterface*, SizedColumnTypes *argTypes*) `[inherited]`

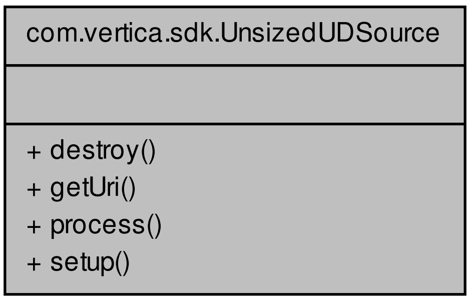
Perform per instance initialization. This function may throw errors.

com.vertica.sdk.UnsizedUDSource Class Reference

Inheritance diagram for com.vertica.sdk.UnsizedUDSource:



Collaboration diagram for com.vertica.sdk.UnsizedUDSource:



Public Member Functions

- void **destroy** (ServerInterface srvInterface) throws UdfException
- String getUri ()
- abstract StreamState **process** (ServerInterface srvInterface, DataBuffer output) throws UdfException
- void **setup** (ServerInterface srvInterface) throws UdfException

Detailed Description

UnsizedUDSource

Base class for UDSource. Used with IterativeSourceFactory if computing the size of a source up front would be prohibitively expensive, or if the number of distinct sources would be prohibitively large to use the standard API.

Not intended or optimized for typical applications.

Member Function Documentation

String com.vertica.sdk.UnsizedUDSource.getUri ( )

UnsizedUDSource::getUri()

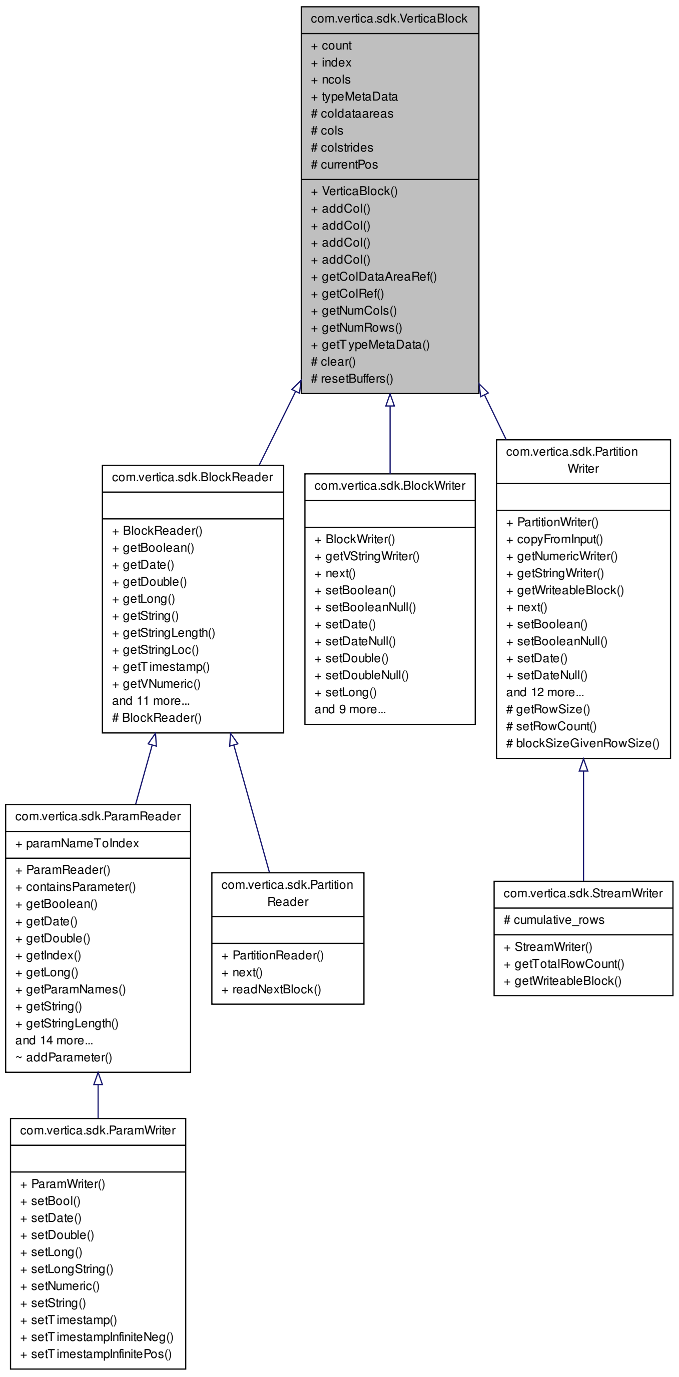
Return the URI of the current source of data.

This function will be invoked during execution to fill in monitoring information.

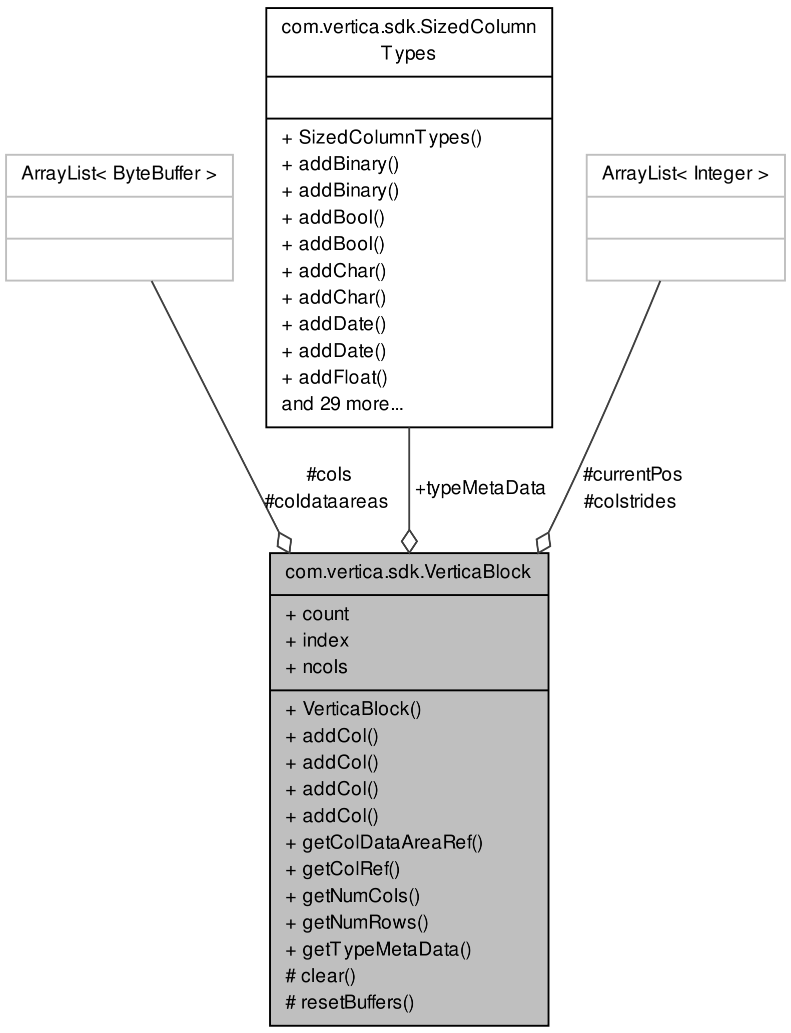
com.vertica.sdk.VerticaBlock Class Reference

: Represents an in-memory block of tuples

Inheritance diagram for com.vertica.sdk.VerticaBlock:



Collaboration diagram for com.vertica.sdk.VerticaBlock:



Public Member Functions

- **VerticaBlock** (int _ncols, int _rowcount)
- void addCol (ByteBuffer arg, int colstride, VerticaType dt, String colName)
- void addCol (ByteBuffer arg, int colstride, VerticaType dt)
- void addCol (ByteBuffer arg, ByteBuffer da, int colstride, VerticaType dt)
- void addCol (ByteBuffer arg, ByteBuffer da, int colstride, VerticaType dt, String colName)
- ByteBuffer getColDataAreaRef (int idx)
- ByteBuffer getColRef (int idx)
- int getNumCols ()
- int getNumRows ()
- SizedColumnTypes getTypeMetaData ()

Public Attributes

- int **count**

- int **index**
- int **ncols**
- SizedColumnTypes **typeMetaData**

Protected Member Functions

- void **clear** ()
- void **resetBuffers** ()

Protected Attributes

- ArrayList< ByteBuffer > **coldataareas**
- ArrayList< ByteBuffer > **cols**
- ArrayList< Integer > **colstrides**
- ArrayList< Integer > **currentPos**

Detailed Description

: Represents an in-memory block of tuples

Member Function Documentation

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, int *colstride*, VerticaType *dt*, String *colName*)

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
colstride	The stride between data instances.
dt	The type of input.
colName	Name of the column

Referenced by com.vertica.sdk.VerticaBlock.addCol().

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, int *colstride*, VerticaType *dt*)

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
colstride	The stride between data instances.
dt	The type of input.

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg*, ByteBuffer *da*, int *colstride*, VerticaType *dt*)

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
da	The location to find out of band string data.
colstride	The stride between data instances.
dt	The type of input.

void com.vertica.sdk.VerticaBlock.addCol (ByteBuffer *arg,* ByteBuffer *da,* int *colstride,* VerticaType *dt,* String *colName*)

Add the location for reading a particular argument.

Parameters

arg	The base location to find data.
da	The location to find out of band string data.
colstride	The stride between data instances.
dt	The type of input.
colName	Name of the column

ByteBuffer com.vertica.sdk.VerticaBlock.getColDataAreaRef (int *idx*)

Get the ByteBuffer that stores out of line string data (Data Area) for the idx'th argument

Parameters

idx	

Returns

Referenced by com.vertica.sdk.BlockReader.getVString().

ByteBuffer com.vertica.sdk.VerticaBlock.getColRef (int *idx*)

Returns

a ByteBuffer to the idx'th argument, containing data for the column

Example:

```
* ByteBuffer a = arg_reader.getColPtr(0);
*
```

Referenced by com.vertica.sdk.PartitionWriter.copyFromInput(), com.vertica.sdk.BlockReader.getBoolean(), com.-vertica.sdk.BlockReader.getDouble(), com.vertica.sdk.BlockReader.getLong(), com.vertica.sdk.BlockReader.-getStringLength(), com.vertica.sdk.BlockReader.getStringLoc(), com.vertica.sdk.BlockReader.getVNumeric(), com.vertica.sdk.BlockReader.getVString(), com.vertica.sdk.BlockWriter.getVStringWriter(), com.vertica.sdk.-BlockReader.isBooleanNull(), com.vertica.sdk.ParamWriter.setBool(), com.vertica.sdk.BlockWriter.setBoolean(), com.vertica.sdk.BlockWriter.setBooleanNull(), com.vertica.sdk.ParamWriter.setDouble(), com.vertica.sdk.Block-Writer.setDouble(), com.vertica.sdk.BlockWriter.setDoubleNull(), com.vertica.sdk.PartitionWriter.setLong(), com.-vertica.sdk.BlockWriter.setLongNull(), com.vertica.sdk.ParamWriter.setLongString(), com.vertica.sdk.BlockWriter.-setNumeric(), com.vertica.sdk.ParamWriter.setNumeric(), com.vertica.sdk.BlockWriter.setString(), com.vertica.-sdk.ParamWriter.setString(), and com.vertica.sdk.BlockWriter.setStringNull().

int com.vertica.sdk.VerticaBlock.getNumCols ( )

Returns

the number of arguments held by this reader.

int com.vertica.sdk.VerticaBlock.getNumRows ( )

Returns

the number of rows held by this block.

SizedColumnTypes com.vertica.sdk.VerticaBlock.getTypeMetaData ( )

Returns

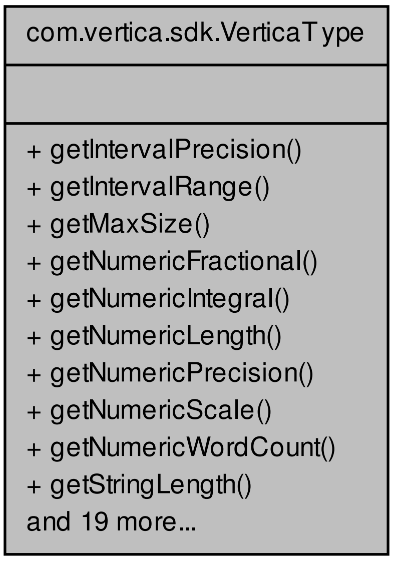
information about the types and numbers of arguments

Referenced by com.vertica.sdk.ParamReader.getType().

com.vertica.sdk.VerticaType Class Reference

Represents types of data that are passed into and returned back from user's code.

Collaboration diagram for com.vertica.sdk.VerticaType:



Public Member Functions

- int getIntervalPrecision ()

  For INTERVAL data types, returns the precision.
- int getIntervalRange ()

For INTERVAL data types, returns the range.

- int getMaxSize ()

  Returns the maximum size, in bytes, of a data element of this type.

- int **getNumericFractional** ()
- int **getNumericIntegral** ()
- int getNumericLength ()

  For NUMERIC data types, returns the number of bytes required to store an element. Calling this with a non-numeric data type can cause a crash.

- int **getNumericPrecision** ()
- int **getNumericScale** ()
- int **getNumericWordCount** ()
- int getStringLength ()

  For VARCHAR/CHAR/VARBINARY/BINARY data types, returns the length of the string.

- int getTimestampPrecision ()

  For TIMESTAMP data types, returns the precision.

- boolean isBinary ()

  Returns true if this type is BINARY, false otherwise.

- boolean isBool ()

  Returns true if this type is BOOLEAN, false otherwise.

- boolean isChar ()

  Returns true if this type is CHAR, false otherwise.

- boolean isDate ()

  Returns true if this type is DATE, false otherwise.

- boolean isFloat ()

  Returns true if this type is FLOAT, false otherwise.

- boolean isInt ()

  Returns true if this type is INTEGER, false otherwise.

- boolean isLongVarbinary ()

  Returns true if this type is LONGVARBINARY, false otherwise.

- boolean isLongVarchar ()

  Returns true if this type is LONGVARCHAR, false otherwise.

- boolean isNumeric ()

  Returns true if this type is NUMERIC, false otherwise.

- boolean isStringType ()

  Return true for VARCHAR/CHAR/VARBINARY/BINARY data types.

- boolean isTimestamp ()

  Returns true if this type is TIMESTAMP, false otherwise.

- boolean isVarbinary ()

  Returns true if this type is VARBINARY, false otherwise.

- boolean isVarchar ()

  Returns true if this type is VARCHAR, false otherwise.

- void setIntervalPrecision (int precision)

  For INTERVAL data types, sets the precision.

- void setIntervalRange (int range)

  For INTERVAL data types, sets the range.

- void setNumericPrecision (int precision)

  For NUMERIC data types, sets the precision.

- void setNumericScale (int scale)

  For NUMERIC data types, sets the scale.

- void setTimestampPrecision (int precision)

  For TIMESTAMP data types, sets the precision.

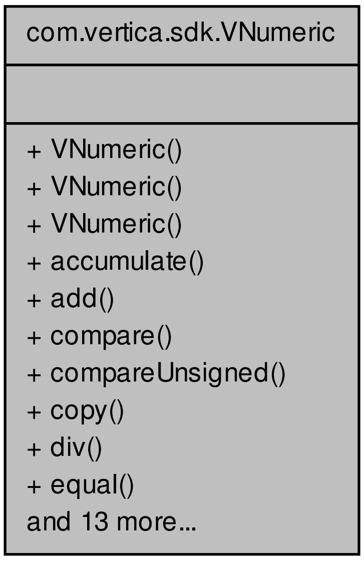
Detailed Description

Represents types of data that are passed into and returned back from user's code.

com.vertica.sdk.VNumeric Class Reference

Representation of NUMERIC, fixed point data types in Vertica.

Collaboration diagram for com.vertica.sdk.VNumeric:



Public Member Functions

- **VNumeric** (ByteBuffer buff, int data_offset, int max_data_len, int typmod)
- **VNumeric** (BigDecimal words, int precision, int scale)
- **VNumeric** (BigDecimal words, int t)
- void **accumulate** (VNumeric from)
- void **add** (VNumeric a, VNumeric b)
- int **compare** (VNumeric from)
- int **compareUnsigned** (VNumeric from)
- void **copy** (VNumeric from)
- void **div** (VNumeric a, VNumeric b)
- boolean **equal** (VNumeric from)
- void **incr** ()
- void **invertSign** ()
- boolean **isNeg** ()
- boolean **isNull** ()
- boolean **isZero** ()
- void **mul** (VNumeric a, VNumeric b)
- void **setNull** ()

- void **setZero** ()
- void **shiftLeft** (int bitsToShift)
- void **shiftRight** (int bitsToShift)
- void **sub** (VNumeric a, VNumeric b)
- void **toString** (ByteBuffer outBuf, int olen)
- String **toString** ()

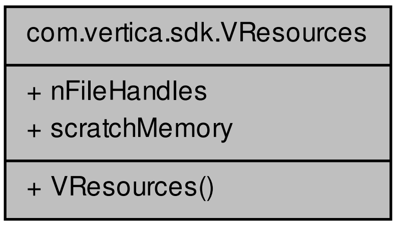
Detailed Description

Representation of NUMERIC, fixed point data types in Vertica.

com.vertica.sdk.VResources Class Reference

Representation of the resources user code can ask Vertica for.

Collaboration diagram for com.vertica.sdk.VResources:



Public Attributes

- int nFileHandles
- long scratchMemory

Detailed Description

Representation of the resources user code can ask Vertica for.

Member Data Documentation

int com.vertica.sdk.VResources.nFileHandles

Number of file handles / sockets required

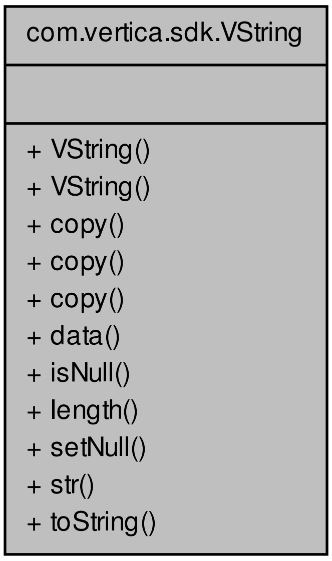
long com.vertica.sdk.VResources.scratchMemory

Amount of RAM in bytes used by User defined function

com.vertica.sdk.VString Class Reference

Representation of a String in Vertica. All character data is internally encoded as UTF-8 characters and is not NULL terminated.

Collaboration diagram for com.vertica.sdk.VString:



Public Member Functions

- VString (ByteBuffer buf, int offset, int total_max_len)

  Contruct a VString object.
- VString (ByteBuffer hbuf, ByteBuffer dbuf, int hoffset, int doffset, int max_dlen)

  Contruct an out of line VString object.
- void copy (byte[] from)

  Copy character data from byte array to the VString's internal buffer.
- void copy (String from)

  Copy character data from String.
- void copy (VString from)

  Copy data from another VString.
- ByteBuffer data ()

  Provides a read-only ByteBuffer to this VString's internal data.
- boolean isNull ()

  Indicates if this VString contains the SQL NULL value.
- int length ()

  Returns the length of this VString.
- void setNull ()

  Sets this VString to the SQL NULL value.
- String str ()

  Provides a copy of this VString's data as a Java String.
- String **toString** ()

Detailed Description

Representation of a String in Vertica. All character data is internally encoded as UTF-8 characters and is not NULL terminated.

Constructor & Destructor Documentation

com.vertica.sdk.VString.VString (ByteBuffer *buf*, int *offset*, int *total_max_len*)

Contruct a VString object.

Parameters

buf	the ByteBuffer providing the space to back the VString
offset	offset of the beginning of VString into the ByteBuffer
total_max_len	the maximum length of the string structure including the header

com.vertica.sdk.VString.VString (ByteBuffer *hbuf*, ByteBuffer *dbuf*, int *hoffset*, int *doffset*, int *max_dlen*)

Contruct an out of line VString object.

Parameters

hbuf	the ByteBuffer with the VString header
dbuf	the ByteBuffer with the actual data
hoffset	of the beginning of VString header in hbuf
doffset	offset of the actual string data
max_dlen	maximum length of the string structure *not* including the header

Member Function Documentation

void com.vertica.sdk.VString.copy (byte[ ] *from*)

Copy character data from byte array to the VString's internal buffer.

Parameters

from	array of bytes input data

void com.vertica.sdk.VString.copy (String *from*)

Copy character data from String.

Parameters

from	Java String object as character input data

void com.vertica.sdk.VString.copy (VString *from*)

Copy data from another VString.

Parameters

from	The source VString

ByteBuffer com.vertica.sdk.VString.data ( )

Provides a read-only ByteBuffer to this VString's internal data.

Returns

the read only character data for this string in a ByteBuffer

Note

The returned string is **not** null terminated

boolean com.vertica.sdk.VString.isNull ( )

Indicates if this VString contains the SQL NULL value.

Returns

true if this string contains the SQL NULL value, false otherwise

Referenced by com.vertica.sdk.VString.str().

int com.vertica.sdk.VString.length ( )

Returns the length of this VString.

Returns

the length of the string, in bytes. Does not include any extra space for null characters.

Referenced by com.vertica.sdk.VString.isNull(), and com.vertica.sdk.VString.str().

String com.vertica.sdk.VString.str ( )

Provides a copy of this VString's data as a Java String.

Returns

a Java String copy of the data in this VString

Index