

Sparksee

5.0.0

Generated by Doxygen 1.5.6

Fri Feb 7 13:09:46 2014

Contents

1	Module Index	1
1.1	Modules	1
2	Class Index	1
2.1	Class Hierarchy	1
3	Class Index	4
3.1	Class List	4
4	Module Documentation	7
4.1	Gdb	7
4.1.1	Typedef Documentation	15
4.1.2	Enumeration Type Documentation	15
4.1.3	Function Documentation	19
4.2	Io	20
4.3	Script	22
4.3.1	Function Documentation	23
4.4	Algorithms	23
5	Class Documentation	25
5.1	AppError Class Reference	25
5.1.1	Detailed Description	26
5.1.2	Constructor & Destructor Documentation	26
5.1.3	Member Function Documentation	26
5.2	Attribute Class Reference	27
5.2.1	Detailed Description	28
5.2.2	Member Function Documentation	28
5.3	AttributeList Class Reference	30
5.3.1	Detailed Description	30
5.3.2	Constructor & Destructor Documentation	30
5.3.3	Member Function Documentation	31
5.4	AttributeListIterator Class Reference	31
5.4.1	Detailed Description	32
5.4.2	Member Function Documentation	32
5.5	AttributeStatistics Class Reference	32
5.5.1	Detailed Description	34
5.5.2	Member Function Documentation	34

5.6	BooleanList Class Reference	36
5.6.1	Detailed Description	37
5.6.2	Constructor & Destructor Documentation	37
5.6.3	Member Function Documentation	37
5.7	BooleanListIterator Class Reference	38
5.7.1	Detailed Description	38
5.7.2	Member Function Documentation	38
5.8	ConnectedComponents Class Reference	39
5.8.1	Detailed Description	39
5.8.2	Constructor & Destructor Documentation	39
5.8.3	Member Function Documentation	40
5.9	Connectivity Class Reference	41
5.9.1	Detailed Description	44
5.9.2	Constructor & Destructor Documentation	44
5.9.3	Member Function Documentation	45
5.10	Context Class Reference	46
5.10.1	Detailed Description	48
5.10.2	Constructor & Destructor Documentation	48
5.10.3	Member Function Documentation	48
5.11	CSVReader Class Reference	50
5.11.1	Detailed Description	51
5.11.2	Member Function Documentation	52
5.12	CSVWriter Class Reference	54
5.12.1	Detailed Description	56
5.12.2	Member Function Documentation	56
5.13	Database Class Reference	58
5.13.1	Detailed Description	59
5.13.2	Member Function Documentation	59
5.14	DatabaseStatistics Class Reference	60
5.14.1	Detailed Description	60
5.14.2	Member Function Documentation	61
5.15	DefaultExport Class Reference	62
5.15.1	Detailed Description	63
5.15.2	Member Function Documentation	63
5.16	EdgeData Class Reference	64
5.16.1	Detailed Description	65

5.16.2 Member Function Documentation	65
5.17 EdgeExport Class Reference	66
5.17.1 Detailed Description	67
5.17.2 Member Function Documentation	67
5.18 EdgeTypeExporter Class Reference	69
5.18.1 Detailed Description	71
5.18.2 Constructor & Destructor Documentation	71
5.18.3 Member Function Documentation	71
5.19 EdgeTypeLoader Class Reference	74
5.19.1 Detailed Description	76
5.19.2 Member Enumeration Documentation	76
5.19.3 Constructor & Destructor Documentation	77
5.19.4 Member Function Documentation	77
5.20 Error Class Reference	80
5.20.1 Detailed Description	81
5.20.2 Constructor & Destructor Documentation	82
5.20.3 Member Function Documentation	82
5.21 Exception Class Reference	82
5.21.1 Detailed Description	83
5.21.2 Constructor & Destructor Documentation	84
5.21.3 Member Function Documentation	84
5.22 ExportManager Class Reference	84
5.22.1 Detailed Description	85
5.22.2 Member Function Documentation	85
5.23 FileNotFoundException Class Reference	88
5.23.1 Detailed Description	89
5.23.2 Constructor & Destructor Documentation	89
5.23.3 Member Function Documentation	89
5.24 Graph Class Reference	89
5.24.1 Detailed Description	94
5.24.2 Member Function Documentation	95
5.25 GraphExport Class Reference	110
5.25.1 Detailed Description	110
5.25.2 Member Function Documentation	111
5.26 Handler< T > Class Template Reference	111
5.26.1 Detailed Description	112

5.27	Int32List Class Reference	113
5.27.1	Detailed Description	113
5.27.2	Constructor & Destructor Documentation	113
5.27.3	Member Function Documentation	114
5.28	Int32ListIterator Class Reference	114
5.28.1	Detailed Description	115
5.28.2	Member Function Documentation	115
5.29	IOError Class Reference	115
5.29.1	Detailed Description	116
5.29.2	Constructor & Destructor Documentation	117
5.29.3	Member Function Documentation	117
5.30	IOException Class Reference	118
5.30.1	Detailed Description	119
5.30.2	Constructor & Destructor Documentation	119
5.30.3	Member Function Documentation	119
5.31	LicenseError Class Reference	119
5.31.1	Detailed Description	121
5.31.2	Constructor & Destructor Documentation	121
5.31.3	Member Function Documentation	121
5.32	NodeExport Class Reference	122
5.32.1	Detailed Description	123
5.32.2	Member Function Documentation	124
5.33	NodeTypeExporter Class Reference	126
5.33.1	Detailed Description	128
5.33.2	Constructor & Destructor Documentation	128
5.33.3	Member Function Documentation	128
5.34	NodeTypeLoader Class Reference	131
5.34.1	Detailed Description	133
5.34.2	Member Enumeration Documentation	133
5.34.3	Constructor & Destructor Documentation	134
5.34.4	Member Function Documentation	134
5.35	NoSuchElementException Class Reference	137
5.35.1	Detailed Description	138
5.35.2	Constructor & Destructor Documentation	138
5.35.3	Member Function Documentation	138
5.36	Objects Class Reference	139

5.36.1 Detailed Description	141
5.36.2 Member Function Documentation	141
5.37 ObjectsIterator Class Reference	146
5.37.1 Detailed Description	147
5.37.2 Member Function Documentation	148
5.38 OIDList Class Reference	148
5.38.1 Detailed Description	149
5.38.2 Constructor & Destructor Documentation	149
5.38.3 Member Function Documentation	149
5.39 OIDListIterator Class Reference	150
5.39.1 Detailed Description	151
5.39.2 Member Function Documentation	151
5.40 Platform Class Reference	151
5.40.1 Detailed Description	151
5.40.2 Member Function Documentation	152
5.41 PlatformStatistics Class Reference	152
5.41.1 Detailed Description	152
5.41.2 Member Function Documentation	153
5.42 RowReader Class Reference	154
5.42.1 Detailed Description	154
5.42.2 Constructor & Destructor Documentation	155
5.42.3 Member Function Documentation	155
5.43 RowWriter Class Reference	156
5.43.1 Detailed Description	156
5.43.2 Constructor & Destructor Documentation	157
5.43.3 Member Function Documentation	157
5.44 ScriptParser Class Reference	157
5.44.1 Detailed Description	158
5.44.2 Member Function Documentation	158
5.45 Session Class Reference	160
5.45.1 Detailed Description	162
5.45.2 Member Function Documentation	162
5.46 ShortestPath Class Reference	162
5.46.1 Detailed Description	165
5.46.2 Constructor & Destructor Documentation	165
5.46.3 Member Function Documentation	165

5.46.4	Member Data Documentation	166
5.47	SinglePairShortestPath Class Reference	167
5.47.1	Detailed Description	169
5.47.2	Constructor & Destructor Documentation	170
5.47.3	Member Function Documentation	170
5.47.4	Member Data Documentation	172
5.48	SinglePairShortestPathBFS Class Reference	172
5.48.1	Detailed Description	175
5.48.2	Constructor & Destructor Documentation	175
5.48.3	Member Function Documentation	175
5.48.4	Member Data Documentation	177
5.49	SinglePairShortestPathDijkstra Class Reference	177
5.49.1	Detailed Description	181
5.49.2	Constructor & Destructor Documentation	181
5.49.3	Member Function Documentation	181
5.49.4	Member Data Documentation	183
5.50	Sparksee Class Reference	184
5.50.1	Detailed Description	185
5.50.2	Constructor & Destructor Documentation	185
5.50.3	Member Function Documentation	185
5.51	SparkseeConfig Class Reference	186
5.51.1	Detailed Description	189
5.51.2	Constructor & Destructor Documentation	191
5.51.3	Member Function Documentation	191
5.52	SparkseeProperties Class Reference	198
5.52.1	Detailed Description	199
5.52.2	Member Function Documentation	199
5.53	SQLException Class Reference	200
5.53.1	Detailed Description	201
5.53.2	Constructor & Destructor Documentation	202
5.53.3	Member Function Documentation	202
5.54	StringList Class Reference	202
5.54.1	Detailed Description	203
5.54.2	Constructor & Destructor Documentation	203
5.54.3	Member Function Documentation	203
5.55	StringListIterator Class Reference	204

5.55.1 Detailed Description	204
5.55.2 Member Function Documentation	205
5.56 StrongConnectivity Class Reference	205
5.56.1 Detailed Description	208
5.56.2 Constructor & Destructor Documentation	209
5.56.3 Member Function Documentation	209
5.57 StrongConnectivityGabow Class Reference	210
5.57.1 Detailed Description	214
5.57.2 Constructor & Destructor Documentation	214
5.57.3 Member Function Documentation	214
5.58 SystemError Class Reference	216
5.58.1 Detailed Description	217
5.58.2 Constructor & Destructor Documentation	218
5.58.3 Member Function Documentation	218
5.59 TextStream Class Reference	218
5.59.1 Detailed Description	220
5.59.2 Constructor & Destructor Documentation	220
5.59.3 Member Function Documentation	220
5.60 Traversal Class Reference	221
5.60.1 Detailed Description	223
5.60.2 Constructor & Destructor Documentation	224
5.60.3 Member Function Documentation	224
5.61 TraversalBFS Class Reference	225
5.61.1 Detailed Description	228
5.61.2 Constructor & Destructor Documentation	228
5.61.3 Member Function Documentation	228
5.62 TraversalDFS Class Reference	230
5.62.1 Detailed Description	232
5.62.2 Constructor & Destructor Documentation	232
5.62.3 Member Function Documentation	233
5.63 Type Class Reference	234
5.63.1 Detailed Description	235
5.63.2 Member Function Documentation	236
5.64 TypeExporter Class Reference	237
5.64.1 Detailed Description	239
5.64.2 Constructor & Destructor Documentation	239

5.64.3	Member Function Documentation	239
5.65	TypeExporterEvent Class Reference	242
5.65.1	Detailed Description	243
5.65.2	Member Function Documentation	243
5.66	TypeExporterListener Class Reference	243
5.66.1	Detailed Description	244
5.66.2	Constructor & Destructor Documentation	244
5.66.3	Member Function Documentation	244
5.67	TypeList Class Reference	244
5.67.1	Detailed Description	245
5.67.2	Constructor & Destructor Documentation	245
5.67.3	Member Function Documentation	246
5.68	TypeListIterator Class Reference	246
5.68.1	Detailed Description	247
5.68.2	Member Function Documentation	247
5.69	TypeLoader Class Reference	247
5.69.1	Detailed Description	249
5.69.2	Member Enumeration Documentation	250
5.69.3	Constructor & Destructor Documentation	250
5.69.4	Member Function Documentation	250
5.70	TypeLoaderEvent Class Reference	254
5.70.1	Detailed Description	255
5.70.2	Member Function Documentation	255
5.71	TypeLoaderListener Class Reference	256
5.71.1	Detailed Description	257
5.71.2	Constructor & Destructor Documentation	257
5.71.3	Member Function Documentation	257
5.72	UnsupportedOperationError Class Reference	257
5.72.1	Detailed Description	258
5.72.2	Constructor & Destructor Documentation	259
5.72.3	Member Function Documentation	259
5.73	Value Class Reference	259
5.73.1	Detailed Description	263
5.73.2	Constructor & Destructor Documentation	263
5.73.3	Member Function Documentation	263
5.74	Values Class Reference	269

5.74.1 Detailed Description	270
5.74.2 Member Function Documentation	271
5.75 ValuesIterator Class Reference	271
5.75.1 Detailed Description	273
5.75.2 Member Function Documentation	273
5.76 WeakConnectivity Class Reference	273
5.76.1 Detailed Description	277
5.76.2 Constructor & Destructor Documentation	277
5.76.3 Member Function Documentation	277
5.77 WeakConnectivityDFS Class Reference	279
5.77.1 Detailed Description	283
5.77.2 Constructor & Destructor Documentation	283
5.77.3 Member Function Documentation	283
5.78 WrongArgumentError Class Reference	285
5.78.1 Detailed Description	287
5.78.2 Constructor & Destructor Documentation	287
5.78.3 Member Function Documentation	287

1 Module Index

1.1 Modules

Here is a list of all modules:

Gdb	7
Io	20
Script	22
Algorithms	23

2 Class Index

2.1 Class Hierarchy

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

Attribute	27
AttributeList	30
AttributeListIterator	31

AttributeStatistics	32
BooleanList	36
BooleanListIterator	38
ConnectedComponents	39
Connectivity	41
StrongConnectivity	205
StrongConnectivityGabow	210
WeakConnectivity	273
WeakConnectivityDFS	279
Context	46
DatabaseStatistics	60
EdgeData	64
EdgeExport	66
Exception	82
Error	80
AppError	25
LicenseError	119
UnsupportedOperationError	257
WrongArgumentError	285
SystemError	216
IOError	115
IOException	118
FileNotFoundException	88
NoSuchElementException	137
SQLException	200
ExportManager	84
DefaultExport	62
GraphExport	110
Handler< T >	111

Handler< sparksee_core::DbGraph >	111
Graph	89
Handler< sparksee_core::GraphPool >	111
Database	58
Handler< sparksee_core::GraphTextStream >	111
TextStream	218
Handler< sparksee_core::Objects >	111
Objects	139
Handler< sparksee_core::Session >	111
Session	160
Handler< sparksee_core::SPARKSEE >	111
Sparksee	184
Handler< sparksee_core::Value >	111
Value	259
Handler< sparksee_core::Values >	111
Values	269
ValuesIterator	271
Int32List	113
Int32ListIterator	114
NodeExport	122
ObjectsIterator	146
OIDList	148
OIDListIterator	150
Platform	151
PlatformStatistics	152
RowReader	154
CSVReader	50
RowWriter	156
CSVWriter	54

ScriptParser	157
ShortestPath	162
SinglePairShortestPath	167
SinglePairShortestPathBFS	172
SinglePairShortestPathDijkstra	177
SparkseeConfig	186
SparkseeProperties	198
StringList	202
StringListIterator	204
Traversal	221
TraversalBFS	225
TraversalDFS	230
Type	234
TypeExporter	237
EdgeTypeExporter	69
NodeTypeExporter	126
TypeExporterEvent	242
TypeExporterListener	243
TypeList	244
TypeListIterator	246
TypeLoader	247
EdgeTypeLoader	74
NodeTypeLoader	131
TypeLoaderEvent	254
TypeLoaderListener	256

3 Class Index

3.1 Class List

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

AppError (Application error class)	25
Attribute (Attribute data class)	27
AttributeList (Sparksee attribute identifier list)	30
AttributeListIterator (AttributeList iterator class)	31
AttributeStatistics (Attribute statistics class)	32
BooleanList (Boolean list)	36
BooleanListIterator (BooleanList iterator class)	38
ConnectedComponents (ConnectedComponents class)	39
Connectivity (Connectivity class)	41
Context (Context class)	46
CSVReader (CSVReader interface)	50
CSVWriter (CSVWriter interface)	54
Database (Database class)	58
DatabaseStatistics (Database statistics)	60
DefaultExport (Default implementation for ExportManager class)	62
EdgeData (Edge data class)	64
EdgeExport (Stores edge exporting values)	66
EdgeTypeExporter (EdgeTypeExporter class)	69
EdgeTypeLoader (EdgeTypeLoader class)	74
Error (Error class)	80
Exception (Exception class)	82
ExportManager (Defines how to export a graph to an external format)	84
FileNotFoundException (File not found exception class)	88
Graph (Graph class)	89
GraphExport (Stores the graph exporting values)	110
Handler< T > (Handles a reference)	111
Int32List (Sparksee 32-bit signed integer list)	113
Int32ListIterator (Int32List iterator class)	114
IOError (IO error class)	115

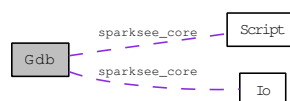
IOException (IO exception class)	118
LicenseError (License error class)	119
NodeExport (Stores the node exporting values)	122
NodeTypeExporter (NodeTypeExporter class)	126
NodeTypeLoader (NodeTypeLoader class)	131
NoSuchElementException (No such element exception class)	137
Objects (Object identifier set class)	139
ObjectsIterator (ObjectsIterator class)	146
OIDList (Sparksee object identifier list)	148
OIDListIterator (OIDList iterator class)	150
Platform (Platform class)	151
PlatformStatistics (Platform data and statistics)	152
RowReader (RowReader interface)	154
RowWriter (RowWriter interface)	156
ScriptParser (ScriptParser)	157
Session (Session class)	160
ShortestPath (ShortestPath class)	162
SinglePairShortestPath (SinglePairShortestPath class)	167
SinglePairShortestPathBFS (SinglePairShortestPathBFS class)	172
SinglePairShortestPathDijkstra (SinglePairShortestPathDijkstra class)	177
Sparksee (Sparksee class)	184
SparkseeConfig (Sparksee configuration class)	186
SparkseeProperties (Sparksee properties file)	198
SQLException (SQL exception class)	200
StringList (String list)	202
StringListIterator (StringList iterator class)	204
StrongConnectivity (StrongConnectivity class)	205
StrongConnectivityGabow (This class can be used to solve the problem of finding strongly connected components in a directed graph)	210
SystemError (System error class)	216

TextStream (TextStream class)	218
Traversal (Traversal class)	221
TraversalBFS (Breadth-First Search implementation of Traversal)	225
TraversalDFS (Depth-First Search (DFS) implementation of Traversal)	230
Type (Type data class)	234
TypeExporter (Base TypeExporter class)	237
TypeExporterEvent (Provides information about the progress of an TypeExproter instance)	242
TypeExporterListener (Interface to be implemented to receive TypeExporterEvent events from a TypeExporter)	243
TypeList (Sparksee type identifier list)	244
TypeListIterator (TypeList iterator class)	246
TypeLoader (Base TypeLoader class)	247
TypeLoaderEvent (Provides information about the progress of a TypeLoader instance)	254
TypeLoaderListener (Interface to be implemented to receive TypeLoaderEvent events from a TypeLoader)	256
UnsupportedOperationError (Unsupported operation error class)	257
Value (Value class)	259
Values (Value set class)	269
ValuesIterator (Values iterator class)	271
WeakConnectivity (WeakConnectivity class)	273
WeakConnectivityDFS (WeakConnectivityDFS class)	279
WrongArgumentError (Wrong argument error class)	285

4 Module Documentation

4.1 Gdb

Collaboration diagram for Gdb:



Files

- file [common.h](#)
It contains common includes and definitions as well as set basic data types and enumerations.
- file [Database.h](#)
It contains the definition of [Database](#) class as well as some other related classes.
- file [Exception.h](#)
It contains a hierarchy of exceptions.
- file [Export.h](#)
It contains the declaration of [ExportManager](#) interface and [GraphExport](#), [NodeExport](#), etc classes.
- file [Graph.h](#)
It contains the definition of [Graph](#) class.
- file [Graph_data.h](#)
It contains the definition of some Graph-related classes.
- file [Handler.h](#)
It contains the definition of [Handler](#) class.
- file [Objects.h](#)
It contains the definition of [Object](#) class.
- file [ObjectsIterator.h](#)
It contains the definition of [ObjectsIterator](#) class.
- file [Session.h](#)
It contains the definition of [Session](#) class.
- file [Sparksee.h](#)
It contains the declaration of [Sparksee](#) and [SparkseeConfig](#) classes.
- file [Stream.h](#)
It contains the definition of stream classes.
- file [Value.h](#)
It contains the definitio of [Value](#) class.
- file [Values.h](#)
It contains the definition of [Values](#) class.
- file [ValuesIterator.h](#)
It contains the definition of [ValuesIterator](#) class.

Classes

- class [PlatformStatistics](#)
Platform data and statistics.
- class [Platform](#)
Platform class.
- class [DatabaseStatistics](#)
Database statistics.
- class [Database](#)
Database class.
- class [Exception](#)
Exception class.
- class [IOException](#)
IO exception class.
- class [FileNotFoundException](#)
File not found exception class.
- class [NoSuchElementException](#)
No such element exception class.
- class [SQLException](#)
SQL exception class.
- class [Error](#)
Error class.
- class [SystemError](#)
System error class.
- class [AppError](#)
Application error class.
- class [WrongArgumentError](#)
Wrong argument error class.
- class [IOError](#)
IO error class.
- class [LicenseError](#)
License error class.
- class [UnsupportedOperationError](#)
Unsupported operation error class.

- class [GraphExport](#)
Stores the graph exporting values.
- class [NodeExport](#)
Stores the node exporting values.
- class [EdgeExport](#)
Stores edge exporting values.
- class [ExportManager](#)
Defines how to export a graph to an external format.
- class [DefaultExport](#)
Default implementation for [ExportManager](#) class.
- class [Graph](#)
Graph class.
- class [Type](#)
Type data class.
- class [TypeList](#)
Sparksee type identifier list.
- class [TypeListIterator](#)
TypeList iterator class.
- class [Attribute](#)
Attribute data class.
- class [AttributeList](#)
Sparksee attribute identifier list.
- class [AttributeListIterator](#)
AttributeList iterator class.
- class [OIDList](#)
Sparksee object identifier list.
- class [OIDListIterator](#)
OIDList iterator class.
- class [AttributeStatistics](#)
Attribute statistics class.
- class [EdgeData](#)
Edge data class.
- class [StringList](#)
String list.

- class [StringListIterator](#)
StringList iterator class.
- class [BooleanList](#)
Boolean list.
- class [BooleanListIterator](#)
BooleanList iterator class.
- class [Int32List](#)
Sparksee 32-bit signed integer list.
- class [Int32ListIterator](#)
Int32List iterator class.
- class [Handler< T >](#)
Handles a reference.
- class [Objects](#)
Object identifier set class.
- class [ObjectsIterator](#)
ObjectsIterator class.
- class [Session](#)
Session class.
- class [SparkseeProperties](#)
Sparksee properties file.
- class [SparkseeConfig](#)
Sparksee configuration class.
- class [Sparksee](#)
Sparksee class.
- class [TextStream](#)
TextStream class.
- class [Value](#)
Value class.
- class [Values](#)
Value set class.
- class [ValuesIterator](#)
Values iterator class.

Defines

- #define `BEGIN_SPARKSEE_NAMESPACE` namespace sparksee {
Beginning macro for the sparksee namespace.
- #define `END_SPARKSEE_NAMESPACE` }
Ending macro for the sparksee namespace.
- #define `BEGIN_SPARKSEE_GDB_NAMESPACE` BEGIN_SPARKSEE_NAMESPACE namespace gdb {
Beginning macro for the sparksee::gdb namespace.
- #define `END_SPARKSEE_GDB_NAMESPACE` END_SPARKSEE_NAMESPACE }
Ending macro for the sparksee::gdb namespace.
- #define `BEGIN_SPARKSEE_IO_NAMESPACE` BEGIN_SPARKSEE_NAMESPACE namespace io {
Beginning macro for the sparksee::io namespace.
- #define `END_SPARKSEE_IO_NAMESPACE` END_SPARKSEE_NAMESPACE }
Ending macro for the sparksee::io namespace.
- #define `BEGIN_SPARKSEE_SCRIPT_NAMESPACE` BEGIN_SPARKSEE_NAMESPACE namespace script {
Beginning macro for the sparksee::script namespace.
- #define `END_SPARKSEE_SCRIPT_NAMESPACE` END_SPARKSEE_NAMESPACE }
Ending macro for the sparksee::script namespace.
- #define `BEGIN_SPARKSEE_ALGORITHMS_NAMESPACE` BEGIN_SPARKSEE_NAMESPACE namespace algorithms {
Beginning macro for the sparksee::algorithms namespace.
- #define `END_SPARKSEE_ALGORITHMS_NAMESPACE` END_SPARKSEE_NAMESPACE }
Ending macro for the sparksee::algorithms namespace.
- #define `BEGIN_SPARKSEE_TEST_NAMESPACE` BEGIN_SPARKSEE_NAMESPACE namespace test {
Beginning macro for the sparksee::test namespace.
- #define `END_SPARKSEE_TEST_NAMESPACE` END_SPARKSEE_NAMESPACE }
Ending macro for the sparksee::test namespace.

Typedefs

- typedef bool `bool_t`
Boolean type.
- typedef char `char_t`

Character type.

- typedef wchar_t [uchar_t](#)
Unicode character type.
- typedef signed int [int32_t](#)
32-bit signed integer type.
- typedef signed long long [int64_t](#)
64-bit signed integer type.
- typedef double [double64_t](#)
64-bit double type.
- typedef [int32_t](#) [type_t](#)
Graph node or edge type type.
- typedef [int32_t](#) [attr_t](#)
Graph attribute type.
- typedef [int64_t](#) [oid_t](#)
Graph OID type.
- typedef [int32_t](#) [ColorRGB](#)
Color codified as RGB 32-bit int.

Enumerations

- enum [ObjectType](#) {
 [Node](#),
 [Edge](#) }
Object type enumeration.
- enum [DataType](#) {
 [Boolean](#),
 [Integer](#),
 [Long](#),
 [Double](#),
 [Timestamp](#),
 [String](#),
 [Text](#),
 [OID](#) }
Data type (domain) enumeration.

- enum `Condition` {
 `Equal`,
 `GreaterEqual`,
 `GreaterThan`,
 `LessEqual`,
 `LessThan`,
 `NotEqual`,
 `Like`,
 `LikeNoCase`,
 `Between`,
 `RegExp` }
 Condition operators enumeration.
- enum `Order` {
 `Ascendent`,
 `Descendent` }
 Order enumeration.
- enum `EdgesDirection` {
 `Ingoing`,
 `Outgoing`,
 `Any` }
 Edges direction enumeration.
- enum `AttributeKind` {
 `Basic`,
 `Indexed`,
 `Unique` }
 Attribute kind enumeration.
- enum `LogLevel` {
 `Off`,
 `Severe`,
 `Warning`,
 `Info`,
 `Config`,
 `Fine`,
 `Debug` }
 Log level enumeration.
- enum `ExportType` {
 `Graphviz`,
 `GraphML`,
 `YGraphML` }

Export type.

- enum `NodeShape` {
`Box`,
`Round` }

Node shape.

Functions

- `std::wostream & operator<< (std::wostream &wostrm, const enum DataType &dt)`
Easy STL printing operator redefinition.
- `std::wostream & operator<< (std::wostream &wostrm, const enum AttributeKind &ak)`
Easy STL printing operator redefinition.
- `Handler::Handler (T *h)`
Creates a new instance with the given reference.
- virtual `Handler::~~Handler ()`
Destructor.
- `T * Handler::GetHandler ()`
Gets the handled reference.
- `const T * Handler::GetHandler () const`
Gets the handled reference.
- `void Handler::SetHandler (T *h)`
Sets the handled reference.
- `void Handler::FreeHandler ()`
Frees (deletes) the handled reference.
- `bool_t Handler::IsNull () const`
Gets if the handler is NULL.

Variables

- `BEGIN_SPARKSEE_GDB_NAMESPACE` typedef unsigned char `byte_t`
Byte type.

4.1.1 Typedef Documentation

4.1.1.1 typedef int32_t ColorRGB

Color codified as RGB 32-bit int.

Bits 24-31 are alpha, 16-23 are red, 8-15 are green, 0-7 are blue.

4.1.2 Enumeration Type Documentation

4.1.2.1 enum AttributeKind

[Attribute](#) kind enumeration.

It determines the indexing-capabilities of an attribute.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

Enumerator:

Basic Basic attribute (non indexed attribute).

Indexed Indexed attribute.

Unique Unique attribute (indexed + unique restriction).

Unique restriction sets two objects cannot have the same value for an attribute but NULL.

4.1.2.2 enum Condition

Condition operators enumeration.

It is mainly used in the attribute-based graph select operations.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

Enumerator:

Equal Equal condition (==).

Null values can be used together with this condition to retrieve all objects having a null value for an attribute.

GreaterEqual Greater or equal condition (>=).

Null values cannot be used together with this condition.

GreaterThan Greater than condition (>).

Null values cannot be used together with this condition.

LessEqual Less or equal condition (<=).

Null values cannot be used together with this condition.

LessThan Less than condition (<).

Null values cannot be used together with this condition.

NotEqual Not equal condition (!=).

Null values can be used together with this condition to retrieve all objects having a non-null value for an attribute.

Like Substring condition.

Null values cannot be used together with this condition.

This condition can just be used together with String values. It allows for searching substrings (case sensitive). Ex:

```
'AAABBBCCCD' Like 'BBB'    returns TRUE
'AAABBBCCCD' Like 'bbb'    returns FALSE
'AAABBBCCCD' Like 'E'      returns FALSE
```

LikeNoCase Substring (no case sensitive) condition.

Null values cannot be used together with this condition.

This condition can just be used together with String values. It allows for searching substrings (no case sensitive). Ex:

```
'AAABBBCCCD' LikeNoCase 'BBB'    returns TRUE
'AAABBBCCCD' LikeNoCase 'bbb'    returns TRUE
'AAABBBCCCD' LikeNoCase 'E'      returns FALSE
```

Between In range operator condition ([x,y]).

Null values cannot be used together with this condition.

RegExp Regular expression condition.

Null values cannot be used together with this condition.

This condition can just be used together with String values.

Regular expression format conforms most of the POSIX Extended Regular Expressions so it is case sensitive.

See the 'Regular expressions' section in the 'SPARKSEE User Manual' for details.

4.1.2.3 enum DataType

Data type (domain) enumeration.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

Enumerator:

Boolean Boolean data type.

Integer 32-bit signed integer data type.

Long 64-bit signed integer data type.

Double 64-bit signed double data type.

Timestamp Distance from Epoch (UTC) time in milliseconds precision.

It just works properly with timestamps in the range ['1970-01-01 00:00:01' UTC, '2038-01-19 03:14:07' UTC].

String Unicode string data type.

2048 characters maximum length.

Text Large unicode character object (CLOB) data type.

See also:

[TextStream](#)

OID Object identifier (OID) data type.

4.1.2.4 enum EdgesDirection

Edges direction enumeration.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

Enumerator:

Ingoing In-going edges.

Outgoing Out-going edges.

Any In-going or out-going edges.

4.1.2.5 enum ExportType

Export type.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

Enumerator:

Graphviz Export to Graphviz format.

Graphviz home page: <http://www.graphviz.org>

GraphML Export to GraphML format.

GraphML home page: <http://graphml.graphdrawing.org/>

YGraphML Export to YGRAPHML format.

It is an GraphML format extended with a set of yWorks ("http://www.yworks.com") extensions. Thus, it allows for the visualization of the exported graph with the public yEd visualization tool ("http://www.yworks.com/products/yed").

4.1.2.6 enum LogLevel

Log level enumeration.

Log level priority order is as follows, from minimum to maximum log information: Off (log is disabled), Severe, Warning, Info, Config, Fine, Debug.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

Enumerator:

Off Disable log.

Severe Severe log level.

This is the lower log level, just errors are shown.

Warning Warning log level.

Errors and warnings are shown.

Info Info log level.

Errors, warnings and information messages are shown.

Config Config log level.

Errors, warnings, information messages and configuration details of the different components are shown.

Fine Fine log level.

This is the higher and finest public log level, everything is dumped to the log.

Debug Debug log level.

This is for [Sparksee](#) development purposes and just works with debug versions of the library.

4.1.2.7 enum NodeShape

Node shape.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

Enumerator:

Box Box shape.

Round Round shape.

4.1.2.8 enum ObjectType

Object type enumeration.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

Enumerator:

Node Node object type.

Edge Edge object type.

4.1.2.9 enum Order

Order enumeration.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

Enumerator:

Ascendent From lower to higher.

Descendent From higher to lower.

4.1.3 Function Documentation

4.1.3.1 `template<typename T> const T* Handler< T >::GetHandler () const [inline, inherited]`

Gets the handled reference.

Returns:

The handled reference.

4.1.3.2 `template<typename T> T* Handler< T >::GetHandler () [inline, inherited]`

Gets the handled reference.

Returns:

The handled reference.

4.1.3.3 `template<typename T> Handler< T >::Handler (T * h)` [inline, inherited]

Creates a new instance with the given reference.

Parameters:

h [in] Reference to be handled.

4.1.3.4 `template<typename T> bool_t Handler< T >::IsNull () const` [inline, protected, inherited]

Gets if the handler is NULL.

Returns:

TRUE if the handler is NULL, FALSE otherwise.

Reimplemented in [TextStream](#), and [Value](#).

4.1.3.5 `std::wostream& operator<< (std::wostream & wostrm, const enum AttributeKind & ak)`

Easy STL printing operator redefinition.

It allows to do: ... << [sparksee::gdb::Basic](#) << ...

4.1.3.6 `std::wostream& operator<< (std::wostream & wostrm, const enum DataType & dt)`

Easy STL printing operator redefinition.

It allows to do: ... << [sparksee::gdb::String](#) << ...

4.1.3.7 `template<typename T> void Handler< T >::SetHandler (T * h)` [inline, protected, inherited]

Sets the handled reference.

Parameters:

h [in] The handled reference.

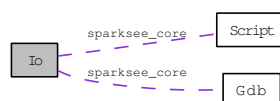
4.1.3.8 `template<typename T> virtual Handler< T >::~~Handler ()` [inline, virtual, inherited]

Destructor.

Frees the handled reference.

4.2 Io

Collaboration diagram for Io:



Files

- file [CSVReader.h](#)
It contains the definition of [CSVReader](#) class.
- file [CSVWriter.h](#)
It contains the definition of [CSVWriter](#) class.
- file [EdgeTypeExporter.h](#)
It contains the definition of [EdgeTypeExporter](#) class.
- file [EdgeTypeLoader.h](#)
It contains the definition of [EdgeTypeLoader](#) class.
- file [NodeTypeExporter.h](#)
It contains the definition of [NodeTypeExporter](#) class.
- file [NodeTypeLoader.h](#)
It contains the definition of [NodeTypeLoader](#) class.
- file [RowReader.h](#)
It contains the definition of [RowReader](#) interface.
- file [RowWriter.h](#)
It contains the definition of [RowWriter](#) interface.
- file [TypeExporter.h](#)
It contains the definition of [TypeExporter](#) classes.
- file [TypeLoader.h](#)
It contains the definition of [TypeLoader](#) classes.

Classes

- class [CSVReader](#)
[CSVReader](#) interface.
- class [CSVWriter](#)
[CSVWriter](#) interface.
- class [EdgeTypeExporter](#)
[EdgeTypeExporter](#) class.
- class [EdgeTypeLoader](#)
[EdgeTypeLoader](#) class.
- class [NodeTypeExporter](#)
[NodeTypeExporter](#) class.

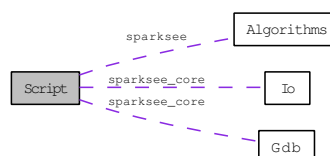
- class [NodeTypeLoader](#)
NodeTypeLoader class.
- class [RowReader](#)
RowReader interface.
- class [RowWriter](#)
RowWriter interface.
- class [TypeExporterEvent](#)
Provides information about the progress of an TypeExproter instance.
- class [TypeExporterListener](#)
Interface to be implemented to receive TypeExporterEvent events from a TypeExporter.
- class [TypeExporter](#)
Base TypeExporter class.
- class [TypeLoaderEvent](#)
Provides information about the progress of a TypeLoader instance.
- class [TypeLoaderListener](#)
Interface to be implemented to receive TypeLoaderEvent events from a TypeLoader.
- class [TypeLoader](#)
Base TypeLoader class.

Functions

- `std::wostream & operator<< (std::wostream &wostrm, const TypeExporterEvent &ev)`
Easy STL printing operator redefinition.
- `std::wostream & operator<< (std::wostream &wostrm, const TypeLoaderEvent &ev)`
Easy STL printing operator redefinition.

4.3 Script

Collaboration diagram for Script:



Files

- file [ScriptParser.h](#)
It contains the declaration of [ScriptParser](#) class.

Classes

- class [ScriptParser](#)
[ScriptParser](#).

Functions

- `std::wostream & operator<< (std::wostream &wostrm, const enum ScriptParserState &state)`
Easy STL printing operator redefinition.

4.3.1 Function Documentation

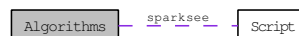
4.3.1.1 `std::wostream& operator<< (std::wostream &wostrm, const enum ScriptParserState &state)`

Easy STL printing operator redefinition.

It allows to do: ... << sparksee::script::SyntaxError << ...

4.4 Algorithms

Collaboration diagram for Algorithms:



Files

- file [ConnectedComponents.h](#)
It contains the definition of [ConnectedComponents](#) class.
- file [Connectivity.h](#)
It contains the definition of [Connectivity](#) class.
- file [Context.h](#)
It contains the definition of [Context](#) class.
- file [ShortestPath.h](#)
It contains the definition of [ShortestPath](#) class.
- file [SinglePairShortestPath.h](#)

It contains the definition of *SinglePairShortestPath* class.

- file [SinglePairShortestPathBFS.h](#)

It contains the definition of *SinglePairShortestPathBFS* class.

- file [SinglePairShortestPathDijkstra.h](#)

It contains the definition of *SinglePairShortestPathDijkstra* class.

- file [StrongConnectivity.h](#)

It contains the definition of *StrongConnectivity* class.

- file [StrongConnectivityGabow.h](#)

It contains the definition of *StrongConnectivityGabow* class.

- file [Traversal.h](#)

It contains the definition of *Traversal* class.

- file [TraversalBFS.h](#)

It contains the definition of *TraversalBFS* class.

- file [TraversalDFS.h](#)

It contains the definition of *TraversalDFS* class.

- file [WeakConnectivity.h](#)

It contains the definition of *WeakConnectivity* class.

- file [WeakConnectivityDFS.h](#)

It contains the definition of *WeakConnectivityDFS* class.

Classes

- class [ConnectedComponents](#)

ConnectedComponents class.

- class [Connectivity](#)

Connectivity class.

- class [Context](#)

Context class.

- class [ShortestPath](#)

ShortestPath class.

- class [SinglePairShortestPath](#)

SinglePairShortestPath class.

- class [SinglePairShortestPathBFS](#)

SinglePairShortestPathBFS class.

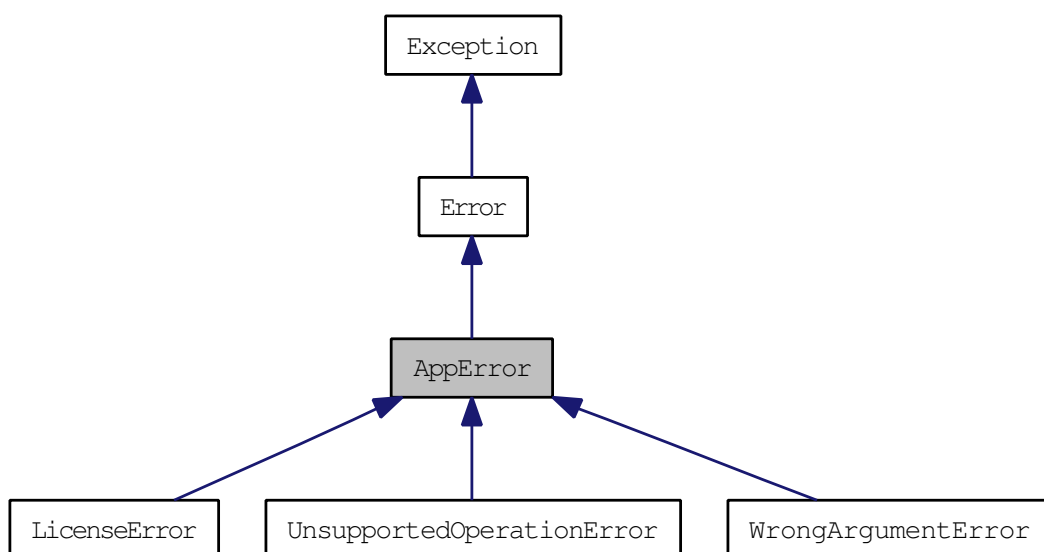
- class [SinglePairShortestPathDijkstra](#)
SinglePairShortestPathDijkstra class.
- class [StrongConnectivity](#)
StrongConnectivity class.
- class [StrongConnectivityGabow](#)
*This class can be used to solve the problem of finding strongly connected components in a **directed** graph.*
- class [Traversal](#)
Traversal class.
- class [TraversalBFS](#)
Breadth-First Search implementation of [Traversal](#).
- class [TraversalDFS](#)
Depth-First Search (DFS) implementation of [Traversal](#).
- class [WeakConnectivity](#)
WeakConnectivity class.
- class [WeakConnectivityDFS](#)
WeakConnectivityDFS class.

5 Class Documentation

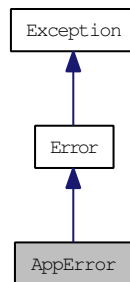
5.1 AppError Class Reference

Application error class.

Inheritance diagram for AppError:



Collaboration diagram for AppError:



Public Member Functions

- [AppError \(\)](#)
Creates a new instance.
- [AppError \(const std::string &mess\)](#)
Creates a new instance.
- virtual [~AppError \(\)](#)
Destructor.
- const std::string & [Message \(\)](#) const
Gets the message of the exception.
- void [SetMessage](#) (const std::string &mess)
Sets the message of the exception.

Static Public Member Functions

- static [Error NewError](#) (int32_t coreErrorCode)
Creates a new [Error](#) instance from a sparksee_core error code.

Protected Attributes

- std::string [message](#)
Message of the exception.

5.1.1 Detailed Description

Application error class.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.1.2 Constructor & Destructor Documentation

5.1.2.1 `AppError::AppError (const std::string & mess)`

Creates a new instance.

Parameters:

mess [in] Message of the exception.

5.1.3 Member Function Documentation

5.1.3.1 `static Error Error::NewError (int32_t coreErrorCode)` [static, inherited]

Creates a new [Error](#) instance from a sparksee_core error code.

Parameters:

coreErrorCode [in] Sparkseecore error code.

Returns:

Depending on the given sparksee_core error, this may return an [Error](#) instance or an specific [Error](#) subclass instance.

5.1.3.2 `const std::string& Exception::Message () const` [inherited]

Gets the message of the exception.

It should be called GetMessage but this is not possible because of a macro defined in Windows.

Returns:

The message of the exception.

5.1.3.3 `void Exception::SetMessage (const std::string & mess)` [inherited]

Sets the message of the exception.

Parameters:

mess [in] Message.

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

- Exception.h

5.2 Attribute Class Reference

[Attribute](#) data class.

Public Member Functions

- [~Attribute](#) ()
Destructor.
- [attr_t GetId](#) () const
Gets the [Sparksee](#) attribute identifier.
- [type_t GetTypeId](#) () const
Gets the [Sparksee](#) type identifier.
- const std::wstring & [GetName](#) () const
Gets the unique attribute name.
- [DataType GetDataType](#) () const
Gets the data type.
- [int64_t GetSize](#) () const
Gets the number of different values.
- [int64_t GetCount](#) () const
Gets the number of non-NULL values.
- [AttributeKind GetKind](#) () const
Gets the attribute kind.
- [bool_t IsSessionAttribute](#) () const
Check if it's a session attribute or a persistent one.

Static Public Attributes

- static const [attr_t InvalidAttribute](#)
Invalid attribute identifier constant.

Friends

- class [Graph](#)

5.2.1 Detailed Description

[Attribute](#) data class.

It contains information about an attribute.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.2.2 Member Function Documentation

5.2.2.1 `attr_t Attribute::GetId () const` [inline]

Gets the [Sparksee](#) attribute identifier.

Returns:

The [Sparksee](#) attribute identifier.

5.2.2.2 `type_t Attribute::GetTypeId () const` [inline]

Gets the [Sparksee](#) type identifier.

Returns:

The [Sparksee](#) type identifier.

5.2.2.3 `const std::wstring& Attribute::GetName () const` [inline]

Gets the unique attribute name.

Returns:

The unique attribute name.

5.2.2.4 `DataType Attribute::GetDataType () const` [inline]

Gets the data type.

Returns:

The DataType.

5.2.2.5 `int64_t Attribute::GetSize () const` [inline]

Gets the number of different values.

Returns:

The number of different values.

5.2.2.6 `int64_t Attribute::GetCount () const` [inline]

Gets the number of non-NULL values.

Returns:

The number of non-NULL values.

5.2.2.7 **AttributeKind** **Attribute::GetKind () const** [inline]

Gets the attribute kind.

Returns:

The **AttributeKind**.

5.2.2.8 **bool_t** **Attribute::IsSessionAttribute () const** [inline]

Check if it's a session attribute or a persistent one.

Returns:

True if it's a session attribute, or false otherwise.

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

- `Graph_data.h`

5.3 **AttributeList** Class Reference

[Sparksee](#) attribute identifier list.

Public Member Functions

- **int32_t** **Count () const**
Number of elements in the list.
- **AttributeListIterator** * **Iterator ()**
Gets a new [AttributeListIterator](#).
- **AttributeList ()**
Constructor.
- **AttributeList** (const std::vector< **attr_t** > &v)
Constructor.
- void **Add** (**attr_t** attr)
Adds a [Sparksee](#) attribute identifier at the end of the list.
- void **Clear ()**
Clears the list.
- **~AttributeList ()**
Destructor.

5.3.1 Detailed Description

[Sparksee](#) attribute identifier list.

It stores a [Sparksee](#) attribute identifier list.

Use [AttributeListIterator](#) to access all elements into this collection.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.3.2 Constructor & Destructor Documentation

5.3.2.1 `AttributeList::AttributeList ()`

Constructor.

This creates an empty list.

5.3.2.2 `AttributeList::AttributeList (const std::vector< attr_t > & v)`

Constructor.

Parameters:

v [in] Vector.

5.3.3 Member Function Documentation

5.3.3.1 `int32_t AttributeList::Count () const` [inline]

Number of elements in the list.

Returns:

Number of elements in the list.

5.3.3.2 `AttributeListIterator* AttributeList::Iterator ()`

Gets a new [AttributeListIterator](#).

Returns:

[AttributeListIterator](#) instance.

5.3.3.3 `void AttributeList::Add (attr_t attr)` [inline]

Adds a [Sparksee](#) attribute identifier at the end of the list.

Parameters:

attr [in] [Sparksee](#) attribute identifier.

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

- `Graph_data.h`

5.4 `AttributeListIterator` Class Reference

`AttributeList` iterator class.

Public Member Functions

- `~AttributeListIterator ()`
Destructor.
- `attr_t Next ()`
Moves to the next element.
- `bool_t HasNext ()`
Gets if there are more elements.

Friends

- class `AttributeList`

5.4.1 Detailed Description

`AttributeList` iterator class.

Iterator to traverse all the `Sparksee` attribute identifier into a `AttributeList` instance.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.4.2 Member Function Documentation

5.4.2.1 `attr_t AttributeListIterator::Next ()` [inline]

Moves to the next element.

Returns:

The next element.

5.4.2.2 `bool_t AttributeListIterator::HasNext ()` [inline]

Gets if there are more elements.

Returns:

TRUE if there are more elements, FALSE otherwise.

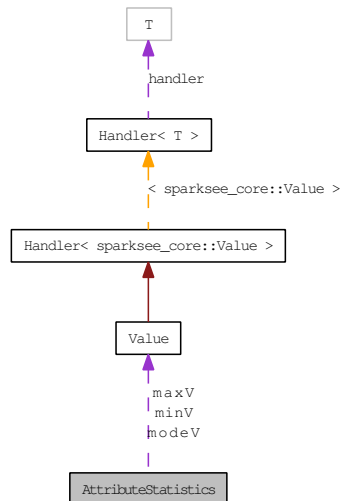
The documentation for this class was generated from the following file:

- `Graph_data.h`

5.5 AttributeStatistics Class Reference

[Attribute](#) statistics class.

Collaboration diagram for AttributeStatistics:



Public Member Functions

- [~AttributeStatistics \(\)](#)
Destructor.
- [int64_t GetTotal \(\) const](#)
Gets the number of objects with a non-NULL [Value](#) (BASIC statistic).
- [int64_t GetNull \(\) const](#)
Gets the number of objects NULL a [Value](#) (BASIC statistics).
- [int64_t GetDistinct \(\) const](#)
Gets the number of distinct values (BASIC statistics).
- [const Value & GetMin \(\) const](#)
Gets the minimum existing value (BASIC statistics).
- [const Value & GetMax \(\) const](#)
Gets the maximum existing value (BASIC statistics).
- [int32_t GetMaxLengthString \(\) const](#)
Gets the maximum length.
- [int32_t GetMinLengthString \(\) const](#)
Gets the minimum length.
- [double64_t GetAvgLengthString \(\) const](#)

Gets the average length.

- `const Value & GetMode () const`
Gets the mode.
- `int64_t GetModeCount () const`
Gets the number of objects with a [Value](#) equal to the mode.
- `double64_t GetMean () const`
Gets the mean or average.
- `double64_t GetVariance () const`
Gets the variance.
- `double64_t GetMedian () const`
Gets the median.

Friends

- class [Graph](#)

5.5.1 Detailed Description

[Attribute](#) statistics class.

It contains statistic data about an attribute.

Some fields are valid just for numerical attributes and others just for string attributes. Also, some statistics are considered BASIC because computing them do not require to traverse all the different values of the attribute. For each getter method the documentation tells if the statistic is BASIC or not. See the [Graph](#) class method `getAttributeStatistics` or check out the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.5.2 Member Function Documentation

5.5.2.1 `int64_t AttributeStatistics::GetTotal () const` [inline]

Gets the number of objects with a non-NULL [Value](#) (BASIC statistic).

Returns:

The number of objects with a non-NULL [Value](#).

5.5.2.2 `int64_t AttributeStatistics::GetNull () const` [inline]

Gets the number of objects NULL a [Value](#) (BASIC statistics).

Returns:

The number of objects NULL a [Value](#).

5.5.2.3 int64_t AttributeStatistics::GetDistinct () const [inline]

Gets the number of distinct values (BASIC statistics).

Returns:

The number of distinct values.

5.5.2.4 const Value& AttributeStatistics::GetMin () const [inline]

Gets the minimum existing value (BASIC statistics).

Returns:

The minimum existing value.

5.5.2.5 const Value& AttributeStatistics::GetMax () const [inline]

Gets the maximum existing value (BASIC statistics).

Returns:

The maximum existing value.

5.5.2.6 int32_t AttributeStatistics::GetMaxLengthString () const [inline]

Gets the maximum length.

If the attribute is not an string attribute, it just returns 0.

Returns:

The maximum length.

5.5.2.7 int32_t AttributeStatistics::GetMinLengthString () const [inline]

Gets the minimum length.

If the attribute is not an string attribute, it just returns 0.

Returns:

The minimum length.

5.5.2.8 double64_t AttributeStatistics::GetAvgLengthString () const [inline]

Gets the average length.

If the attribute is not an string attribute, it just returns 0.

Returns:

The average length.

5.5.2.9 `const Value& AttributeStatistics::GetMode () const` [inline]

Gets the mode.

Mode: Most frequent [Value](#).

Returns:

The mode.

5.5.2.10 `int64_t AttributeStatistics::GetModeCount () const` [inline]

Gets the number of objects with a [Value](#) equal to the mode.

Returns:

The number of objects with a [Value](#) equal to the mode.

5.5.2.11 `double64_t AttributeStatistics::GetMean () const` [inline]

Gets the mean or average.

Mean or average: Sum of all [Values](#) divided by the number of observations.

It is computed just for numerical attributes.

Returns:

The mean.

5.5.2.12 `double64_t AttributeStatistics::GetVariance () const` [inline]

Gets the variance.

It is computed just for numerical attributes.

Returns:

The variance.

5.5.2.13 `double64_t AttributeStatistics::GetMedian () const` [inline]

Gets the median.

Median: Middle value that separates the higher half from the lower.

If $a < b < c$, then the median of the list $\{a, b, c\}$ is b , and if $a < b < c < d$, then the median of the list $\{a, b, c, d\}$ is the mean of b and c , i.e. it is $(b + c)/2$

It is computed just for numerical attributes.

Returns:

The median.

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

- `Graph_data.h`

5.6 BooleanList Class Reference

Boolean list.

Public Member Functions

- `int32_t Count () const`
Number of elements in the list.
- `BooleanListIterator * Iterator ()`
Gets a new [BooleanListIterator](#).
- `BooleanList ()`
Constructor.
- `BooleanList (const std::vector< bool_t > &v)`
Constructor.
- `~BooleanList ()`
Destructor.
- `void Add (sparksee::gdb::bool_t b)`
Adds a Boolean at the end of the list.
- `void Clear ()`
Clears the list.

5.6.1 Detailed Description

Boolean list.

It stores a Boolean list.

Use [BooleanListIterator](#) to access all elements into this collection.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.6.2 Constructor & Destructor Documentation

5.6.2.1 BooleanList::BooleanList ()

Constructor.

This creates an empty list.

5.6.2.2 BooleanList::BooleanList (const std::vector< bool_t > &v)

Constructor.

Parameters:

v [in] Vector.

5.6.3 Member Function Documentation

5.6.3.1 `int32_t BooleanList::Count () const` [inline]

Number of elements in the list.

Returns:

Number of elements in the list.

5.6.3.2 `BooleanListIterator* BooleanList::Iterator ()`

Gets a new [BooleanListIterator](#).

Returns:

[BooleanListIterator](#) instance.

5.6.3.3 `void BooleanList::Add (sparksee::gdb::bool_t b)` [inline]

Adds a Boolean at the end of the list.

Parameters:

b [in] Boolean.

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

- `Graph_data.h`

5.7 BooleanListIterator Class Reference

[BooleanList](#) iterator class.

Public Member Functions

- [~BooleanListIterator \(\)](#)
Destructor.
- [sparksee::gdb::bool_t Next \(\)](#)
Moves to the next element.
- [bool_t HasNext \(\)](#)
Gets if there are more elements.

Friends

- class [BooleanList](#)

5.7.1 Detailed Description

[BooleanList](#) iterator class.

Iterator to traverse all the strings into a [BooleanList](#) instance.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.7.2 Member Function Documentation

5.7.2.1 `sparksee::gdb::bool_t BooleanListIterator::Next ()` `[inline]`

Moves to the next element.

Returns:

The next element.

5.7.2.2 `bool_t BooleanListIterator::HasNext ()` `[inline]`

Gets if there are more elements.

Returns:

TRUE if there are more elements, FALSE otherwise.

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

- Graph_data.h

5.8 ConnectedComponents Class Reference

[ConnectedComponents](#) class.

Public Member Functions

- [ConnectedComponents](#) (sparksee::gdb::Session &s, const std::wstring &materializedattribute)
Creates a new instance of [ConnectedComponents](#).
- virtual [~ConnectedComponents](#) ()
Destructor.
- [sparksee::gdb::int64_t GetConnectedComponent](#) (sparksee::gdb::oid_t idNode)
Returns the connected component where the given node belongs to.
- [sparksee::gdb::int64_t GetCount](#) ()
Returns the number of connected components found in the graph.
- [sparksee::gdb::Objects * GetNodes](#) (sparksee::gdb::int64_t idConnectedComponent)
Returns the collection of nodes contained in the given connected component.

- `sparksee::gdb::int64_t GetSize (sparksee::gdb::int64_t idConnectedComponent)`

Returns the number of nodes contained in the given connected component.

5.8.1 Detailed Description

`ConnectedComponents` class.

This class contains the results processed on a `Connectivity` algorithm.

These results contain information related to the connected components found. We must consider that each connected component has a number in order to identify it. These number identifiers are values from 0 to N-1, where N is the number of different connected components found.

When executing any implementation of the `Connectivity`, it is possible to indicate whether the results of the execution must be stored persistently using the class `Connectivity` `setMaterializedAttribute` method. In case the results are set to be materialized, users can retrieve this data whenever they want, even if the graph has been closed and opened again, just by creating a new instance of this class.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.8.2 Constructor & Destructor Documentation

5.8.2.1 `ConnectedComponents::ConnectedComponents (sparksee::gdb::Session & s, const std::wstring & materializedattribute)`

Creates a new instance of `ConnectedComponents`.

This constructor method can only be called when a previous execution of any implementation of the `Connectivity` class has materialized the results in a common attribute type for all the nodes in the graph. For further information about materializing the results processed on any `Connectivity` execution see the documentation of the `Connectivity#SetMaterializedAttribute` method.

Parameters:

- `s` [in] `Session` to get the graph `Graph` on which the information will be retrieved just by getting the values contained in the given common attribute type for all the nodes in the graph and processing them.
- `materializedattribute` [in] The common attribute type for all the nodes in the graph where data will be retrieved in order to process the results related to the connected components found in the graph.

5.8.3 Member Function Documentation

5.8.3.1 `sparksee::gdb::int64_t ConnectedComponents::GetConnectedComponent (sparksee::gdb::oid_t idNode)`

Returns the connected component where the given node belongs to.

Parameters:

- `idNode` [in] The node identifier for which the connected component identifier where it belongs will be returned.

Returns:

The connected component identifier where the given node identifier belongs to.

5.8.3.2 sparksee::gdb::int64_t ConnectedComponents::GetCount ()

Returns the number of connected components found in the graph.

Returns:

The number of connected components found in the graph.

5.8.3.3 sparksee::gdb::Objects* ConnectedComponents::GetNodes (sparksee::gdb::int64_t *idConnectedComponent*)

Returns the collection of nodes contained in the given connected component.

Parameters:

← *idConnectedComponent* The connected component for which the collection of nodes contained in it will be returned.

Returns:

The collection of node identifiers contained in the given connected component.

5.8.3.4 sparksee::gdb::int64_t ConnectedComponents::GetSize (sparksee::gdb::int64_t *idConnectedComponent*)

Returns the number of nodes contained in the given connected component.

Parameters:

← *idConnectedComponent* The connected component for which the number of nodes contained in it will be returned.

Returns:

The number of nodes contained in the given connected component.

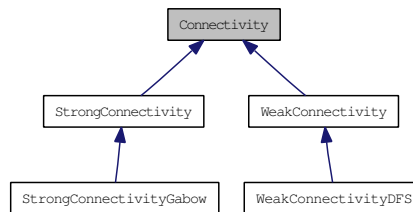
The documentation for this class was generated from the following file:

- ConnectedComponents.h

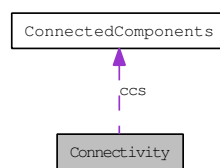
5.9 Connectivity Class Reference

[Connectivity](#) class.

Inheritance diagram for Connectivity:



Collaboration diagram for Connectivity:



Public Member Functions

- virtual `~Connectivity ()`
Destructor.
- virtual void `AddNodeType (sparksee::gdb::type_t t)`
Allows connectivity through nodes of the given type.
- virtual void `AddAllNodeTypes ()`
Allows connectivity through all node types of the graph.
- virtual void `ExcludeNodes (sparksee::gdb::Objects &nodes)`
Set which nodes can't be used.
- virtual void `ExcludeEdges (sparksee::gdb::Objects &edges)`
Set which edges can't be used.
- `ConnectedComponents * GetConnectedComponents ()`
Returns the results generated by the execution of the algorithm.
- virtual void `Run ()=0`
Runs the algorithm in order to find the connected components.
- void `SetMaterializedAttribute (const std::wstring &attributeName)`
Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Protected Types

- `typedef std::map< sparksee::gdb::type_t, sparksee::gdb::EdgesDirection > EdgeTypes_t`
A type definition to store allowed edge types.
- `typedef std::vector< sparksee::gdb::type_t > NodeTypes_t`
A type definition to store allowed node types.

Protected Member Functions

- [Connectivity](#) ([sparksee::gdb::Session](#) &s)
Creates a new instance of [Connectivity](#).
- `void AddEdgeType (sparksee::gdb::type_t t, sparksee::gdb::EdgesDirection d)`
Allows connectivity through edges of the given type.
- `void AddAllEdgeTypes (sparksee::gdb::EdgesDirection d)`
Allows connectivity through all edge types of the graph.
- `void AssertAddedEdges ()`
Check that edges had been added.
- `void AssertAddedNodes ()`
Check that nodes had been added.
- `void AssertNotComputed ()`
Check that the connectivity had not been calculated yet.
- `void SetConnectedComponent (sparksee::gdb::oid_t idNode)`
Assigns the current component to the given node.
- `void SetNodesNotVisited ()`
Set all the selected nodes in [nodesNotVisited](#).
- `void AssertNotComponentAttribute (const std::wstring &attributeName)`
Check that the given attribute name is not already in use.
- `void AssertComputed ()`
Check that the connectivity had been calculated.
- `void AssertEdgeType (sparksee::gdb::type_t edgetype)`
Check that the given edge type is valid.
- `void AssertNodeType (sparksee::gdb::type_t nodetype)`
Check that the given node type is valid.
- `void CreateGlobalPersistentAttribute (const std::wstring &attributeName)`
Set a new persistent global attribute to store the connectivity information.

- void [CreateGlobalTransientAttribute](#) ()
Set a new temporary global attribute to store the connectivity information.
- void [RemoveGlobalAttribute](#) ()
Removes the global attribute where the connectivity information is stored.
- [sparksee::gdb::bool_t IsNodeTypeAllowed](#) ([sparksee::gdb::oid_t](#) nodeId)
Check if the given node has an allowed type.
- [sparksee::gdb::bool_t IsNodeExcluded](#) ([sparksee::gdb::oid_t](#) node)
Check if the given node is forbidden.
- [sparksee::gdb::bool_t IsEdgeExcluded](#) ([sparksee::gdb::oid_t](#) edge)
Check if the given edge is forbidden.

Protected Attributes

- [sparksee::gdb::Session](#) * [sess](#)
Session.
- [sparksee::gdb::Graph](#) * [graph](#)
Graph.
- [EdgeTypes_t](#) [edgeTypes](#)
Allowed edge types.
- [std::vector< sparksee::gdb::type_t >](#) [nodeTypes](#)
Allowed node types.
- [sparksee::gdb::attr_t](#) [attrComponent](#)
common attribute where the connected component information is stored.
- [std::wstring](#) [attrComponentName](#)
name of the common attribute where the connected component information is stored.
- [sparksee::gdb::int64_t](#) [actualComponent](#)
Current component identifier.
- [sparksee::gdb::Objects](#) * [nodesNotVisited](#)
Identifiers of the nodes not visited.
- [sparksee::gdb::bool_t](#) [matResults](#)
Materialized results.
- [sparksee::gdb::bool_t](#) [computed](#)
True if the connectivity has been calculated.
- [sparksee::gdb::Objects](#) * [excludedNodes](#)

The set of excluded nodes.

- `sparksee::gdb::Objects * excludedEdges`

The set of excluded edges.

- `ConnectedComponents * ccs`

The calculated connectivity information.

5.9.1 Detailed Description

[Connectivity](#) class.

Any class implementing this abstract class can be used to solve a problem related to graph connectivity as finding the strongly connected components or finding the weakly connected components.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.9.2 Constructor & Destructor Documentation

5.9.2.1 `Connectivity::Connectivity (sparksee::gdb::Session & s)` `[protected]`

Creates a new instance of [Connectivity](#).

Parameters:

`s` [in] [Session](#) to get the graph from and calculate the connectivity

5.9.3 Member Function Documentation

5.9.3.1 `virtual void Connectivity::ExcludeNodes (sparksee::gdb::Objects & nodes)` `[virtual]`

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters:

`nodes` [in] A set of node identifiers that must be kept intact until the destruction of the class.

5.9.3.2 `virtual void Connectivity::ExcludeEdges (sparksee::gdb::Objects & edges)` `[virtual]`

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters:

`edges` [in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.9.3.3 ConnectedComponents* Connectivity::GetConnectedComponents ()

Returns the results generated by the execution of the algorithm.

These results contain information related to the connected components found as the number of different components, the set of nodes contained in each component or many other data.

Returns:

Returns an instance of the class [ConnectedComponents](#) which contain information related to the connected components found.

5.9.3.4 virtual void Connectivity::Run () [pure virtual]

Runs the algorithm in order to find the connected components.

This method can be called only once.

Implemented in [StrongConnectivityGabow](#), and [WeakConnectivityDFS](#).

5.9.3.5 void Connectivity::SetMaterializedAttribute (const std::wstring & attributeName)

Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it is only necessary to create a new instance of the class [ConnectedComponents](#) indicating the graph and the name of the common attribute type which stores the results. This instance will have all the information related to the connected components found in the moment of the execution of the algorithm that stored this data.

It is possible to run the algorithm without specifying this parameter in order to avoid materializing the results of the execution.

Parameters:

attributeName [in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

5.9.3.6 void Connectivity::AddEdgeType (sparksee::gdb::type_t t, sparksee::gdb::EdgesDirection d) [protected]

Allows connectivity through edges of the given type.

Parameters:

t [in] Edge type.

d [in] Edge direction.

Reimplemented in [StrongConnectivity](#).

5.9.3.7 void Connectivity::AddAllEdgeTypes (sparksee::gdb::EdgesDirection d) [protected]

Allows connectivity through all edge types of the graph.

Parameters:

d [in] Edge direction.

Reimplemented in [StrongConnectivity](#).

5.9.3.8 void Connectivity::SetNodesNotVisited () [protected]

Set all the selected nodes in nodesNotVisited.

That's all the nodes of the allowed node types but not the excluded ones.

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

- Connectivity.h

5.10 Context Class Reference

[Context](#) class.

Public Member Functions

- void [AddEdgeType](#) (sparksee::gdb::type_t t, sparksee::gdb::EdgesDirection d)
Allows for traversing edges of the given type.
- void [AddAllEdgeTypes](#) (sparksee::gdb::EdgesDirection d)
Allows for traversing all edge types of the graph.
- void [AddNodeType](#) (sparksee::gdb::type_t t)
Allows for traversing nodes of the given type.
- void [AddAllNodeTypes](#) ()
Allows for traversing all node types of the graph.
- void [ExcludeNodes](#) (sparksee::gdb::Objects &nodes)
Set which nodes can't be used.
- void [ExcludeEdges](#) (sparksee::gdb::Objects &edges)
Set which edges can't be used.
- sparksee::gdb::Objects * [Compute](#) ()
Gets the resulting collection of nodes.
- void [SetMaximumHops](#) (sparksee::gdb::int32_t maxhops, sparksee::gdb::bool_t include)
Sets the maximum hops restriction.
- virtual [~Context](#) ()
Destructor.
- [Context](#) (sparksee::gdb::Session &s, sparksee::gdb::oid_t node)
Creates a new instance.

Static Public Member Functions

- static sparksee::gdb::Objects * [Compute](#) (sparksee::gdb::Session &s, [sparksee::gdb::oid_t](#) node, [sparksee::gdb::TypeList](#) *nodeTypes, [sparksee::gdb::TypeList](#) *edgeTypes, [sparksee::gdb::EdgesDirection](#) dir, [sparksee::gdb::int32_t](#) maxhops, [sparksee::gdb::bool_t](#) include)

Helper method to easily compute a context from a node.

Protected Attributes

- sparksee::gdb::Session * [sess](#)
Session.
- sparksee::gdb::Graph * [graph](#)
Graph.
- [sparksee::gdb::oid_t](#) src
Source node of the traversal.
- std::map< [sparksee::gdb::type_t](#), [sparksee::gdb::EdgesDirection](#) > [edgeTypes](#)
Allowed edge types.
- std::vector< [sparksee::gdb::type_t](#) > [nodeTypes](#)
Allowed node types.
- [sparksee::gdb::int32_t](#) maxHops
Maximum number of hops allowed.
- [sparksee::gdb::bool_t](#) inclusive
Include those nodes at distance <= maxhops or just those nodes at distance == maxhops.
- sparksee::gdb::Objects * [excludedNodes](#)
The set of excluded nodes.
- sparksee::gdb::Objects * [excludedEdges](#)
The set of excluded edges.

5.10.1 Detailed Description

[Context](#) class.

It provides a very similar functionality than the [Traversal](#) classes. The main difference is [Context](#) returns a resulting collection whereas [Traversal](#) provides an iterator behaviour.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.10.2 Constructor & Destructor Documentation

5.10.2.1 Context::Context (sparksee::gdb::Session & *s*, sparksee::gdb::oid_t *node*)

Creates a new instance.

Parameters:

- s* [in] [Session](#) to get the graph from and perform operation.
- node* [in] Node to start traversal from.

5.10.3 Member Function Documentation

5.10.3.1 void Context::AddEdgeType (sparksee::gdb::type_t *t*, sparksee::gdb::EdgesDirection *d*)

Allows for traversing edges of the given type.

Parameters:

- t* [in] Edge type.
- d* [in] Edge direction.

5.10.3.2 void Context::AddAllEdgeTypes (sparksee::gdb::EdgesDirection *d*)

Allows for traversing all edge types of the graph.

Parameters:

- d* [in] Edge direction.

5.10.3.3 void Context::ExcludeNodes (sparksee::gdb::Objects & *nodes*)

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters:

- nodes* [in] A set of node identifiers that must be kept intact until the destruction of the class.

5.10.3.4 void Context::ExcludeEdges (sparksee::gdb::Objects & *edges*)

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters:

- edges* [in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.10.3.5 sparksee::gdb::Objects* Context::Compute ()

Gets the resulting collection of nodes.

Returns:

The resulting collection of nodes.

5.10.3.6 void Context::SetMaximumHops (sparksee::gdb::int32_t *maxhops*, sparksee::gdb::bool_t *include*)

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

Parameters:

maxhops [in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

include [in] If TRUE, the resulting collection will include those nodes at distance less or equal than the given one, otherwise it will just include those nodes at distance equal than the given one. This parameter just makes sense if maxhops is different from 0; in that case it includes all nodes no matters the distance.

5.10.3.7 static sparksee::gdb::Objects* Context::Compute (sparksee::gdb::Session & *s*, sparksee::gdb::oid_t *node*, sparksee::gdb::TypeList * *nodeTypes*, sparksee::gdb::TypeList * *edgeTypes*, sparksee::gdb::EdgesDirection *dir*, sparksee::gdb::int32_t *maxhops*, sparksee::gdb::bool_t *include*) [static]

Helper method to easily compute a context from a node.

Parameters:

s [in] [Session](#) to get the graph from and perform operation.

node [in] Node to start traversal from.

nodeTypes [in] Allowed node type list. NULL means all node types are allowed.

edgeTypes [in] Allowed edge type list. NULL means all edge types are allowed.

dir [in] Allowed direction for the allowed edge types.

maxhops [in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

include [in] If TRUE, the resulting collection will include those nodes at distance less or equal than the given one, otherwise it will just include those nodes at distance equal than the given one. This parameter just makes sense if maxhops is different from 0; in that case it includes all nodes no matters the distance.

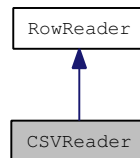
The documentation for this class was generated from the following file:

- Context.h

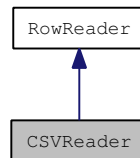
5.11 CSVReader Class Reference

[CSVReader](#) interface.

Inheritance diagram for CSVReader:



Collaboration diagram for CSVReader:



Public Member Functions

- [CSVReader](#) ()
Constructs [CSVReader](#).
- void [SetSeparator](#) (const std::wstring &sep) throw (sparksee::gdb::Error)
Sets the character used to separate fields in the file.
- void [SetQuotes](#) (const std::wstring "es) throw (sparksee::gdb::Error)
Sets the character used to quote fields.
- void [SetMultilines](#) (sparksee::gdb::int32_t numExtralines)
Allows the use of fields with more than one line.
- void [SetSingleLine](#) ()
Only allows single line fields.
- void [SetStartLine](#) (sparksee::gdb::int32_t startLine)
Sets the number of lines to be skiped from the beginning.
- void [SetNumLines](#) (sparksee::gdb::int32_t numLines)
Used to limit the number of lines that will be read.
- void [SetLocale](#) (const std::wstring &localeStr)
Sets the locale that will be used to read the file.
- void [Open](#) (const std::wstring &f) throw (sparksee::gdb::IOException)
Opens the source file path.

- `sparksee::gdb::bool_t Reset ()` throw (`sparksee::gdb::IOException`)
Moves the reader to the beginning.
- `sparksee::gdb::bool_t Read (sparksee::gdb::StringList &row)` throw (`sparksee::gdb::IOException`)
Reads the next row as a string array.
- `sparksee::gdb::int32_t GetRow ()` throw (`sparksee::gdb::IOException`)
The row number for the current row.
- `void Close ()` throw (`sparksee::gdb::IOException`)
Closes the reader.
- `virtual ~CSVReader ()`
Destructor.

5.11.1 Detailed Description

`CSVReader` interface.

A very simple CSV reader.

It works as any other `RowReader`, but `open` must be called once before the first read operation.

Using the format [RFC 4180](#).

Except: leading and trailing spaces, adjacent to CSV separator character, are trimmed.

You can use your own separators and quote characters. By default the separator is the comma (,) and the quote character is the double quotes (").

Fields with multiple lines can be allowed (and the maximum lines specified), but the default is a single line.

The locale string can be used to set the language, country and the file encoding. The format must be "[language_territory][.codeset]". But only the file encoding is being used in the current version.

The languages supported are: "en_US", "es_ES" and "ca_ES".

The file encodings supported are: "utf8" and "iso88591".

For example:

To don't change the default locales, use an empty string: "".

To read a file in utf8 with the default language settings use ".utf8".

To read a file in iso88591 with English language use: "en_US.iso88591".

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.11.2 Member Function Documentation

5.11.2.1 `void CSVReader::SetSeparator (const std::wstring & sep)` throw (`sparksee::gdb::Error`)

Sets the character used to separate fields in the file.

Parameters:

sep [in] Separator character.

5.11.2.2 void CSVReader::SetQuotes (const std::wstring & *quotes*) throw (sparksee::gdb::Error)

Sets the character used to quote fields.

Parameters:

quotes [in] Quote character.

5.11.2.3 void CSVReader::SetMultilines (sparksee::gdb::int32_t *numExtralines*)

Allows the use of fields with more than one line.

Parameters:

numExtralines [in] Maximum number of extra lines for each column (0==unlimited, N==N+1 total rows).

5.11.2.4 void CSVReader::SetStartLine (sparksee::gdb::int32_t *startLine*)

Sets the number of lines to be skipped from the beginning.

Parameters:

startLine [in] The line number to skip for start reading

5.11.2.5 void CSVReader::SetNumLines (sparksee::gdb::int32_t *numLines*)

Used to limit the number of lines that will be read.

Parameters:

numLines [in] The maximum number of lines to read (0 == unlimited)

5.11.2.6 void CSVReader::SetLocale (const std::wstring & *localeStr*)

Sets the locale that will be used to read the file.

Parameters:

localeStr [in] The locale string for the file encoding.

5.11.2.7 void CSVReader::Open (const std::wstring &f) throw (sparksee::gdb::IOException)

Opens the source file path.

Parameters:

f [in] CSV file path.

Exceptions:

IOException If bad things happen opening the file.

5.11.2.8 sparksee::gdb::bool_t CSVReader::Reset () throw (sparksee::gdb::IOException)
[virtual]

Moves the reader to the beginning.

Restarts the reader.

Returns:

true if the reader can be restarted, false otherwise.

Exceptions:

IOException If bad things happen during the restart.

Implements [RowReader](#).

5.11.2.9 sparksee::gdb::bool_t CSVReader::Read (sparksee::gdb::StringList &row) throw (sparksee::gdb::IOException) [virtual]

Reads the next row as a string array.

Parameters:

row [out] A string list with each comma-separated element as a separate entry.

Returns:

Returns true if a row had been read or false otherwise.

Exceptions:

IOException If bad things happen during the read.

Implements [RowReader](#).

5.11.2.10 sparksee::gdb::int32_t CSVReader::GetRow () throw (sparksee::gdb::IOException)
[virtual]

The row number for the current row.

Returns:

The current row number; 0 if there is no current row.

Exceptions:

[*IOException*](#) If it fails.

Implements [RowReader](#).

5.11.2.11 void CSVReader::Close () throw (sparksee::gdb::IOException) [virtual]

Closes the reader.

Exceptions:

[*IOException*](#) If the close fails.

Implements [RowReader](#).

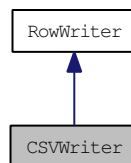
The documentation for this class was generated from the following file:

- CSVReader.h

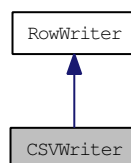
5.12 CSVWriter Class Reference

[CSVWriter](#) interface.

Inheritance diagram for CSVWriter:



Collaboration diagram for CSVWriter:

**Public Member Functions**

- [CSVWriter \(\)](#)
Creates a new instance.
- void [SetSeparator](#) (const std::wstring &sep) throw (sparksee::gdb::Error)
Sets the character used to separate fields in the file.
- void [SetQuotes](#) (const std::wstring "es) throw (sparksee::gdb::Error)

Sets the character used to quote fields.

- void [SetAutoQuotes](#) (sparksee::gdb::bool_t autoquotes)
Sets on/off the automatic quote mode.
- void [SetForcedQuotes](#) (sparksee::gdb::BooleanList &forcequotes)
Disables the automatic quote mode and forces to be quoted those positions set to TRUE in the given vector.
- void [SetLocale](#) (const std::wstring &localeStr)
Sets the locale that will be used to write the file.
- void [Open](#) (const std::wstring &f) throw (sparksee::gdb::IOException)
Opens the output file path.
- void [Write](#) (sparksee::gdb::StringList &row) throw (sparksee::gdb::IOException, sparksee::gdb::Error)
Writes the next row.
- void [Close](#) () throw (sparksee::gdb::IOException, sparksee::gdb::Error)
Closes the writer.
- virtual [~CSVWriter](#) ()
Destructor.

5.12.1 Detailed Description

[CSVWriter](#) interface.

A very simple CSV writer implementing [RowWriter](#).

It works as any other [RowWriter](#), but open must be called once before the first write operation.

It uses the format RFC 4180: <http://tools.ietf.org/html/rfc4180>

You can use your own separators and quote characters. By default the separator is the comma (,) and the quote character is the double quotes (") and autoquote is enabled.

See the [CSVReader](#) locale documentation or the SPARKSEE User Manual.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.12.2 Member Function Documentation

5.12.2.1 void CSVWriter::SetSeparator (const std::wstring & sep) throw (sparksee::gdb::Error)

Sets the character used to separate fields in the file.

Parameters:

sep [in] Separator character.

5.12.2.2 void CSVWriter::SetQuotes (const std::wstring & *quotes*) throw (sparksee::gdb::Error)

Sets the character used to quote fields.

Parameters:

quotes [in] Quote character.

5.12.2.3 void CSVWriter::SetAutoQuotes (sparksee::gdb::bool_t *autoquotes*)

Sets on/off the automatic quote mode.

If there are forced quotes, setting autoquotes on will clear them. If the autoquotes is set to off and no forced quotes are provided, there will not be any quote.

Parameters:

autoquotes [in] If TRUE it enables the automatic quote mode, if FALSE it disables it.

5.12.2.4 void CSVWriter::SetForcedQuotes (sparksee::gdb::BooleanList & *forcequotes*)

Disables the automatic quote mode and forces to be quoted those positions set to TRUE in the given vector.

Parameters:

forcequotes [in] A booleanList with the position for each column that must be quoted set to true.

5.12.2.5 void CSVWriter::SetLocale (const std::wstring & *localeStr*)

Sets the locale that will be used to write the file.

Parameters:

localeStr [in] The locale string for the file encoding.

5.12.2.6 void CSVWriter::Open (const std::wstring & *f*) throw (sparksee::gdb::IOException)

Opens the output file path.

Parameters:

f [in] Output file path.

Exceptions:

IOException If bad things happen opening the file.

5.12.2.7 `void CSVWriter::Write (sparksee::gdb::StringList & row) throw (sparksee::gdb::IOException, sparksee::gdb::Error) [virtual]`

Writes the next row.

Parameters:

row [in] Row of data.

Exceptions:

IOException If bad things happen during the write.

Implements [RowWriter](#).

5.12.2.8 `void CSVWriter::Close () throw (sparksee::gdb::IOException, sparksee::gdb::Error) [virtual]`

Closes the writer.

Exceptions:

IOException If the close fails.

Implements [RowWriter](#).

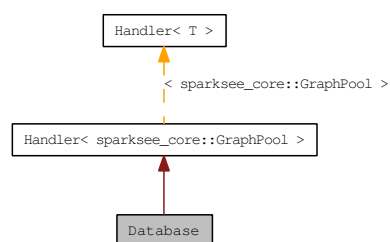
The documentation for this class was generated from the following file:

- CSVWriter.h

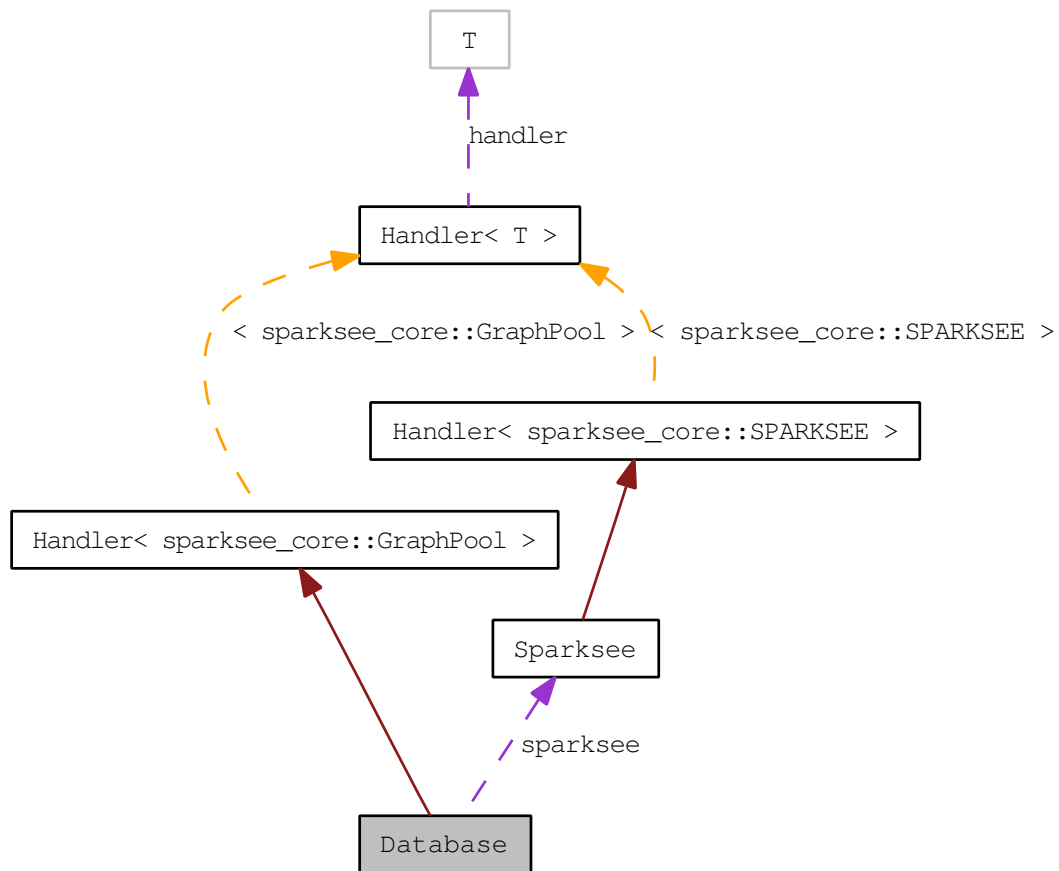
5.13 Database Class Reference

[Database](#) class.

Inheritance diagram for Database:



Collaboration diagram for Database:



Public Member Functions

- virtual `~Database ()`
Destructor.
- const std::wstring & `GetAlias ()` const
Gets the alias of the `Database`.
- const std::wstring & `GetPath ()` const
Gets the path of the `Database`.
- `Session * NewSession ()`
Creates a new `Session`.
- void `GetStatistics (DatabaseStatistics &stats)`
Gets `Database` statistics.

Friends

- class [Sparksee](#)
- class [Graph](#)

5.13.1 Detailed Description

[Database](#) class.

All the data of the [Database](#) is stored into a persistent file which just can be created or open through a [Sparksee](#) instance.

Also, all the manipulation of a [Database](#) must be done by means of a [Session](#) which can be initiated from a [Database](#) instance.

Multiple Databases do not share the memory, that is there is no negotiation among them. In those cases, memory must be prefixed for each [Database](#). To do that, use the SPARKSEConfig.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.13.2 Member Function Documentation

5.13.2.1 `const std::wstring& Database::GetAlias () const` [inline]

Gets the alias of the [Database](#).

Returns:

The alias of the [Database](#).

5.13.2.2 `const std::wstring& Database::GetPath () const` [inline]

Gets the path of the [Database](#).

Returns:

The path of the [Database](#).

5.13.2.3 `void Database::GetStatistics (DatabaseStatistics & stats)`

Gets [Database](#) statistics.

Parameters:

stats [out] The [DatabaseStatistics](#) instance.

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

- Database.h

5.14 DatabaseStatistics Class Reference

[Database](#) statistics.

Public Member Functions

- [int64_t GetRead \(\) const](#)
Gets total read data in KBytes.
- [int64_t GetWrite \(\) const](#)
Gets total written data in KBytes.
- [int64_t GetData \(\) const](#)
Gets database size in KBytes.
- [int64_t GetCache \(\) const](#)
Gets cache size in KBytes.
- [int64_t GetTemp \(\) const](#)
Gets temporary storage file size in KBytes.
- [int64_t GetSessions \(\) const](#)
Gets the number of sessions.

Friends

- class [Database](#)

5.14.1 Detailed Description

[Database](#) statistics.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.14.2 Member Function Documentation

5.14.2.1 [int64_t DatabaseStatistics::GetRead \(\) const](#) [inline]

Gets total read data in KBytes.

Returns:

Total read data in KBytes.

5.14.2.2 [int64_t DatabaseStatistics::GetWrite \(\) const](#) [inline]

Gets total written data in KBytes.

Returns:

Total read written in KBytes.

5.14.2.3 int64_t DatabaseStatistics::GetData () const [inline]

Gets database size in KBytes.

Returns:

[Database](#) size in KBytes.

5.14.2.4 int64_t DatabaseStatistics::GetCache () const [inline]

Gets cache size in KBytes.

Returns:

Cache size in KBytes.

5.14.2.5 int64_t DatabaseStatistics::GetTemp () const [inline]

Gets temporary storage file size in KBytes.

Returns:

Temporary storage file size in KBytes.

5.14.2.6 int64_t DatabaseStatistics::GetSessions () const [inline]

Gets the number of sessions.

Returns:

The number of sessions.

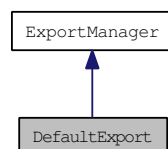
The documentation for this class was generated from the following file:

- Database.h

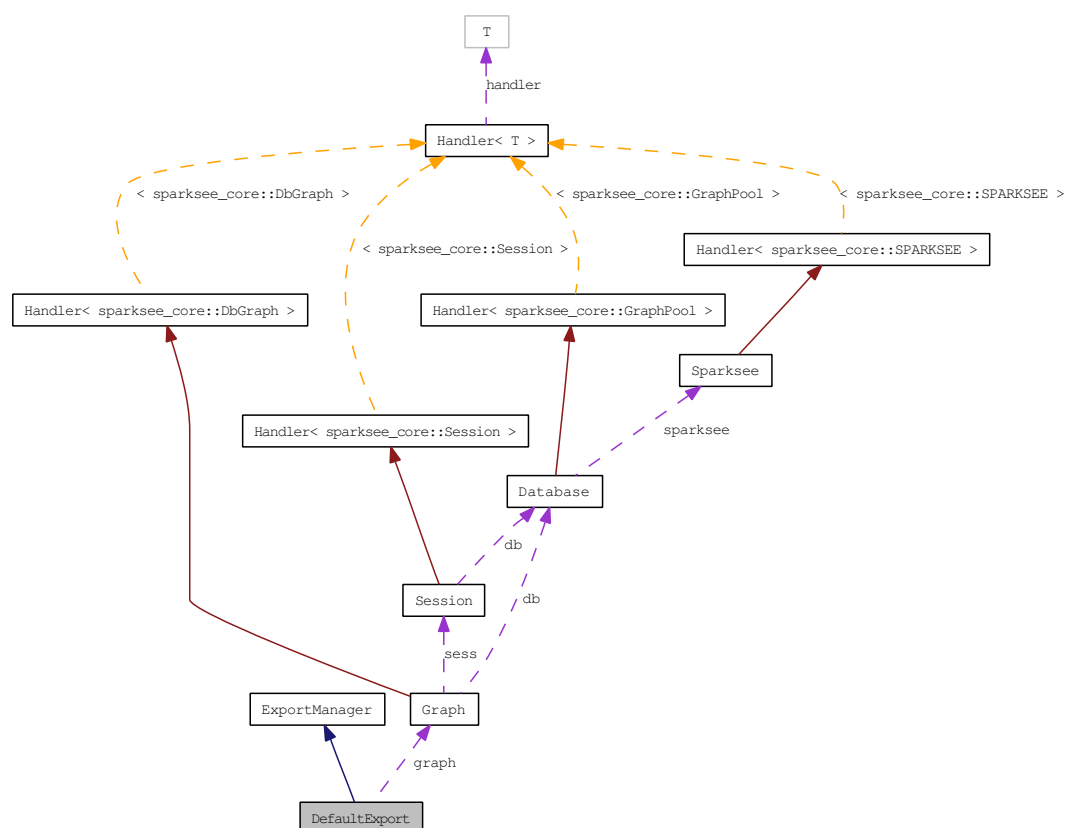
5.15 DefaultExport Class Reference

Default implementation for [ExportManager](#) class.

Inheritance diagram for DefaultExport:



Collaboration diagram for DefaultExport:



Public Member Functions

- [DefaultExport \(\)](#)
Creates a new instance.
- virtual [~DefaultExport \(\)](#)
Destructor.
- void [Prepare \(Graph *graph\)](#)
Default implementation of the [ExportManager](#) class method Prepare.
- void [Release \(\)](#)
Default implementation of the [ExportManager](#) class method Release.
- [bool_t GetGraph \(GraphExport &graphExport\)](#)
Default implementation of the [ExportManager](#) class method GetGraph.
- [bool_t GetNodeType \(type_t type, NodeExport &nodeExport\)](#)
Default implementation of the [ExportManager](#) class method GetNodeType.
- [bool_t GetEdgeType \(type_t type, EdgeExport &edgeExport\)](#)

Default implementation of the [ExportManager](#) class method [GetEdgeType](#).

- **bool_t GetNode** (oid_t node, [NodeExport](#) &nodeExport)

Default implementation of the [ExportManager](#) class method [GetNode](#).

- **bool_t GetEdge** (oid_t edge, [EdgeExport](#) &edgeExport)

Default implementation of the [ExportManager](#) class method [GetEdge](#).

- **bool_t EnableType** (type_t type)

Default implementation of the [ExportManager](#) class method [EnableType](#).

5.15.1 Detailed Description

Default implementation for [ExportManager](#) class.

It uses the default values from [GraphExport](#), [NodeExport](#) and [EdgeExport](#) to export all node and edge types.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.15.2 Member Function Documentation

5.15.2.1 bool_t DefaultExport::GetGraph ([GraphExport](#) & *graphExport*) [virtual]

Default implementation of the [ExportManager](#) class method [GetGraph](#).

This sets the default [GraphExport](#) values and "Graph" as the label.

Implements [ExportManager](#).

5.15.2.2 bool_t DefaultExport::GetNodeType (type_t type, [NodeExport](#) & *nodeExport*) [virtual]

Default implementation of the [ExportManager](#) class method [GetNodeType](#).

This sets de default [NodeExport](#) values.

Implements [ExportManager](#).

5.15.2.3 bool_t DefaultExport::GetEdgeType (type_t type, [EdgeExport](#) & *edgeExport*) [virtual]

Default implementation of the [ExportManager](#) class method [GetEdgeType](#).

This sets de default [EdgeExport](#) values.

Implements [ExportManager](#).

5.15.2.4 bool_t DefaultExport::GetNode (oid_t node, [NodeExport](#) & *nodeExport*) [virtual]

Default implementation of the [ExportManager](#) class method [GetNode](#).

This sets the default [NodeExport](#) values and sets the OID as the label.

Returns:

TRUE.

Implements [ExportManager](#).

5.15.2.5 bool_t DefaultExport::GetEdge (oid_t edge, EdgeExport & edgeExport) [virtual]

Default implementation of the [ExportManager](#) class method GetEdge.

This sets the default [EdgeExport](#) values and sets the OID as the label. Also, it exports the edge as directed just if the edge is directed.

Returns:

TRUE.

Implements [ExportManager](#).

5.15.2.6 bool_t DefaultExport::EnableType (type_t type) [virtual]

Default implementation of the [ExportManager](#) class method EnableType.

This enables all node and edge types to be exported.

Returns:

TRUE.

Implements [ExportManager](#).

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

- [Export.h](#)

5.16 EdgeData Class Reference

Edge data class.

Public Member Functions

- [~EdgeData \(\)](#)
Destructor.
- [oid_t GetEdge \(\) const](#)
Gets the edge identifier.
- [oid_t GetTail \(\) const](#)
Gets the tail of the edge.
- [oid_t GetHead \(\) const](#)
Gets the head of the edge.

Friends

- class [Graph](#)

5.16.1 Detailed Description

Edge data class.

It stores the tail and the head of an edge instance.

In case of undirected edges, the tail and the head are just the two ends of the edge.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.16.2 Member Function Documentation

5.16.2.1 `oid_t EdgeData::GetEdge () const` [inline]

Gets the edge identifier.

Returns:

The [Sparksee](#) edge identifier.

5.16.2.2 `oid_t EdgeData::GetTail () const` [inline]

Gets the tail of the edge.

Returns:

The [Sparksee](#) edge identifier of the tail of the edge.

5.16.2.3 `oid_t EdgeData::GetHead () const` [inline]

Gets the head of the edge.

Returns:

The [Sparksee](#) edge identifier of the head of the edge.

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

- `Graph_data.h`

5.17 EdgeExport Class Reference

Stores edge exporting values.

Public Member Functions

- [EdgeExport](#) ()
Creates a new instance.
- virtual [~EdgeExport](#) ()
Destructor.
- void [SetDefaults](#) ()
Sets to default values.
- const std::wstring & [GetLabel](#) () const
Gets the edge label.
- void [SetLabel](#) (const std::wstring &l)
Sets the edge label.
- bool_t [AsDirected](#) () const
Gets if the edge should be managed as directed.
- void [SetAsDirected](#) (bool_t b)
Sets if the edge should be managed as directed.
- ColorRGB [GetColorRGB](#) () const
Gets the edge color.
- void [SetColorRGB](#) (ColorRGB c)
Sets the edge color.
- ColorRGB [GetLabelColorRGB](#) () const
Gets the edge label color.
- void [SetLabelColorRGB](#) (ColorRGB c)
Sets the edge label color.
- int32_t [GetWidth](#) () const
Gets the edge width.
- void [SetWidth](#) (int32_t w)
Sets the edge width.
- int32_t [GetFontSize](#) () const
Gets the edge label font size.
- void [SetFontSize](#) (int32_t s)
Sets the edge label font size.

5.17.1 Detailed Description

Stores edge exporting values.

Some properties may be ignored depending on the exportation type.

Default values are:

Label: "" (empty string).

As directed: TRUE.

Color: 13882323 (0xD3D3D3, Light gray).

Label color: 0 (0x000000, Black).

Width: 5px.

Font size: 10.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.17.2 Member Function Documentation

5.17.2.1 `const std::wstring& EdgeExport::GetLabel () const` [inline]

Gets the edge label.

Returns:

The edge label.

5.17.2.2 `void EdgeExport::SetLabel (const std::wstring & l)` [inline]

Sets the edge label.

Parameters:

l [in] The edge label.

5.17.2.3 `bool_t EdgeExport::AsDirected () const` [inline]

Gets if the edge should be managed as directed.

TRUE is the default value. If TRUE, use as directed, otherwise use as undirected.

Returns:

The edge direction.

5.17.2.4 `void EdgeExport::SetAsDirected (bool_t b)` [inline]

Sets if the edge should be managed as directed.

Parameters:

b [in] If TRUE, use as directed, otherwise use as undirected.

5.17.2.5 ColorRGB EdgeExport::GetColorRGB () const [inline]

Gets the edge color.

Returns:

The edge color.

5.17.2.6 void EdgeExport::SetColorRGB (ColorRGB *c*) [inline]

Sets the edge color.

Parameters:

c [in] The edge color.

5.17.2.7 ColorRGB EdgeExport::GetLabelColorRGB () const [inline]

Gets the edge label color.

Returns:

The edge label color.

5.17.2.8 void EdgeExport::SetLabelColorRGB (ColorRGB *c*) [inline]

Sets the edge label color.

Parameters:

c [in] The edge label color.

5.17.2.9 int32_t EdgeExport::GetWidth () const [inline]

Gets the edge width.

Returns:

The edge width.

5.17.2.10 void EdgeExport::SetWidth (int32_t *w*) [inline]

Sets the edge width.

Parameters:

w [in] The edge width.

5.17.2.11 int32_t EdgeExport::GetFontSize () const [inline]

Gets the edge label font size.

Returns:

The edge label font size.

5.17.2.12 void EdgeExport::SetFontSize (int32_t s) [inline]

Sets the edge label font size.

Parameters:

s [in] The edge label font size.

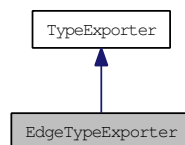
The documentation for this class was generated from the following file:

- Export.h

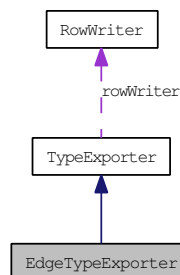
5.18 EdgeTypeExporter Class Reference

[EdgeTypeExporter](#) class.

Inheritance diagram for EdgeTypeExporter:



Collaboration diagram for EdgeTypeExporter:



Public Member Functions

- [EdgeTypeExporter \(\)](#)
Creates a new instance.

- [EdgeTypeExporter](#) ([RowWriter](#) &rw, [sparksee::gdb::Graph](#) &g, [sparksee::gdb::type_t](#) t, [sparksee::gdb::AttributeList](#) &attrs, [sparksee::gdb::int32_t](#) hPos, [sparksee::gdb::int32_t](#) tPos, [sparksee::gdb::attr_t](#) hAttr, [sparksee::gdb::attr_t](#) tAttr)
Creates a new instance.
- virtual [~EdgeTypeExporter](#) ()
Destructor.
- void [Run](#) () throw ([sparksee::gdb::IOException](#), [sparksee::gdb::Error](#))
See the [TypeExporter](#) class [Run](#) method.
- void [SetHeadAttribute](#) ([sparksee::gdb::attr_t](#) attr)
Sets the attribute that will be used to get the value to be dumped for the head of the edge.
- void [SetHeadPosition](#) ([sparksee::gdb::int32_t](#) pos)
Sets the position (index column) of the head attribute in the exported data.
- void [SetTailAttribute](#) ([sparksee::gdb::attr_t](#) attr)
Sets the attribute that will be used to get the value to be dumped for the tail of the edge.
- void [SetTailPosition](#) ([sparksee::gdb::int32_t](#) pos)
Sets the position (index column) of the tail attribute in the exported data.
- void [Register](#) ([TypeExporterListener](#) &tel)
Registers a new listener.
- void [SetRowWriter](#) ([RowWriter](#) &rw)
Sets the output data destination.
- void [SetGraph](#) ([sparksee::gdb::Graph](#) &g)
Sets the graph that will be exported.
- void [SetType](#) ([sparksee::gdb::type_t](#) t)
Sets the type to be exported.
- void [SetAttributes](#) ([sparksee::gdb::AttributeList](#) &attrs)
Sets the list of Attributes.
- void [SetFrequency](#) ([sparksee::gdb::int32_t](#) freq)
Sets the frequency of listener notification.
- void [SetHeader](#) ([sparksee::gdb::bool_t](#) header)
Sets the presence of a header row.

Protected Member Functions

- [sparksee::gdb::bool_t CanRun](#) ()
Checks that all the required settings are ready to run.

- void [NotifyListeners](#) (const [TypeExporterEvent](#) &ev)
Notifies progress to all registered listeners.
- void [RunProcess](#) () throw (sparksee::gdb::IOException, sparksee::gdb::Error)
Runs export process.

5.18.1 Detailed Description

[EdgeTypeExporter](#) class.

Specific [TypeExporter](#) implementation for edge types.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.18.2 Constructor & Destructor Documentation

5.18.2.1 [EdgeTypeExporter::EdgeTypeExporter](#) ([RowWriter](#) & *rw*, [sparksee::gdb::Graph](#) & *g*, [sparksee::gdb::type_t](#) *t*, [sparksee::gdb::AttributeList](#) & *attrs*, [sparksee::gdb::int32_t](#) *hPos*, [sparksee::gdb::int32_t](#) *tPos*, [sparksee::gdb::attr_t](#) *hAttr*, [sparksee::gdb::attr_t](#) *tAttr*) [inline]

Creates a new instance.

Parameters:

- rw* [in] Output [RowWriter](#).
- g* [in] [Graph](#).
- t* [in] [Type](#) identifier.
- attrs* [in] [Attribute](#) identifiers to be exported.
- hPos* [in] The position (index column) for the head value.
- tPos* [in] The position (index column) for the tail value.
- hAttr* [in] The attribute identifier to get the value to be dumped for the head.
- tAttr* [in] The attribute identifier to get the value to be dumped for the tail.

5.18.3 Member Function Documentation

5.18.3.1 void [EdgeTypeExporter::SetHeadAttribute](#) ([sparksee::gdb::attr_t](#) *attr*) [inline]

Sets the attribute that will be used to get the value to be dumped for the head of the edge.

Parameters:

- attr* [in] Head [Attribute](#)

Reimplemented from [TypeExporter](#).

5.18.3.2 void EdgeTypeExporter::SetHeadPosition (sparksee::gdb::int32_t *pos*) [inline]

Sets the position (index column) of the head attribute in the exported data.

Parameters:

pos [in] Head position

Reimplemented from [TypeExporter](#).

5.18.3.3 void EdgeTypeExporter::SetTailAttribute (sparksee::gdb::attr_t *attr*) [inline]

Sets the attribute that will be used to get the value to be dumped for the tail of the edge.

Parameters:

attr [in] Tail [Attribute](#)

Reimplemented from [TypeExporter](#).

5.18.3.4 void EdgeTypeExporter::SetTailPosition (sparksee::gdb::int32_t *pos*) [inline]

Sets the position (index column) of the tail attribute in the exported data.

Parameters:

pos [in] Tail position

Reimplemented from [TypeExporter](#).

5.18.3.5 sparksee::gdb::bool_t TypeExporter::CanRun () [protected, inherited]

Checks that all the required settings are ready to run.

Returns:

Returns true if all the settings are ready.

5.18.3.6 void TypeExporter::NotifyListeners (const TypeExporterEvent & *ev*) [protected, inherited]

Notifies progress to all registered listeners.

Parameters:

ev [in] Progress event to be notified.

5.18.3.7 void TypeExporter::RunProcess () throw (sparksee::gdb::IOException, sparksee::gdb::Error) [protected, inherited]

Runs export process.

Exceptions:

[IOException](#) If bad things happen writting to the [RowWriter](#).

5.18.3.8 void TypeExporter::Register (TypeExporterListener & *tel*) [inherited]

Registers a new listener.

Parameters:

tel [in] [TypeExporterListener](#) to be registered.

5.18.3.9 void TypeExporter::SetRowWriter (RowWriter & *rw*) [inherited]

Sets the output data destination.

Parameters:

rw [in] Input [RowWriter](#).

5.18.3.10 void TypeExporter::SetGraph (sparksee::gdb::Graph & *g*) [inherited]

Sets the graph that will be exported.

Parameters:

g [in] [Graph](#).

5.18.3.11 void TypeExporter::SetType (sparksee::gdb::type_t *t*) [inherited]

Sets the type to be exported.

Parameters:

t [in] [Type](#) identifier.

5.18.3.12 void TypeExporter::SetAttributes (sparksee::gdb::AttributeList & *attrs*) [inherited]

Sets the list of Attributes.

Parameters:

attrs [in] [Attribute](#) identifiers to be exported

5.18.3.13 void TypeExporter::SetFrequency (sparksee::gdb::int32_t *freq*) [inherited]

Sets the frequency of listener notification.

freq [in] Frequency in number of rows managed to notify progress to all listeners

5.18.3.14 void TypeExporter::SetHeader (sparksee::gdb::bool_t header) [inherited]

Sets the presence of a header row.

Parameters:

header [in] If TRUE, a header row is dumped with the name of the attributes.

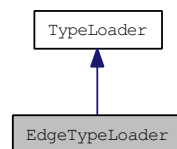
The documentation for this class was generated from the following file:

- EdgeTypeExporter.h

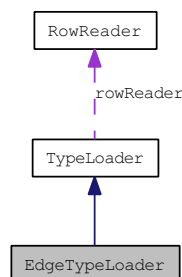
5.19 EdgeTypeLoader Class Reference

[EdgeTypeLoader](#) class.

Inheritance diagram for EdgeTypeLoader:



Collaboration diagram for EdgeTypeLoader:

**Public Member Functions**

- [EdgeTypeLoader](#) ()
Creates a new instance.
- [EdgeTypeLoader](#) ([RowReader](#) &rr, sparksee::gdb::Graph &g, [sparksee::gdb::type_t](#) t, sparksee::gdb::AttributeList &attrs, sparksee::gdb::Int32List &attrsPos, [sparksee::gdb::int32_t](#) hPos, [sparksee::gdb::int32_t](#) tPos, [sparksee::gdb::attr_t](#) hAttr, [sparksee::gdb::attr_t](#) tAttr)
Creates a new instance.
- virtual [~EdgeTypeLoader](#) ()
Destructor.
- void [Run](#) () throw (sparksee::gdb::IOException, sparksee::gdb::Error)

See the [TypeLoader](#) class [Run](#) method.

- void [RunTwoPhases](#) () throw (sparksee::gdb::IOException, sparksee::gdb::Error)
See the [TypeLoader](#) class [RunTwoPhases](#) method.
- void [RunNPhases](#) (sparksee::gdb::int32_t partitions) throw (sparksee::gdb::IOException, sparksee::gdb::Error)
See the [TypeLoader](#) class [RunNPhases](#) method.
- void [SetHeadAttribute](#) (sparksee::gdb::attr_t attr)
Sets the attribute that will be used to find the head of the edge.
- void [SetHeadPosition](#) (sparksee::gdb::int32_t pos)
Sets the position of the head attribute in the source data.
- void [SetTailAttribute](#) (sparksee::gdb::attr_t attr)
Sets the attribute that will be used to find the tail of the edge.
- void [SetTailPosition](#) (sparksee::gdb::int32_t pos)
Sets the position of the tail attribute in the source data.
- void [SetLogError](#) (const std::wstring &path) throw (sparksee::gdb::IOException)
Sets a log error file.
- void [SetLogOff](#) ()
Turns off all the error reporting.
- void [Register](#) (TypeLoaderListener &tel)
Registers a new listener.
- void [SetRowReader](#) (RowReader &rr)
Sets the input data source.
- void [SetGraph](#) (sparksee::gdb::Graph &g)
Sets the graph where the data will be loaded.
- void [SetLocale](#) (const std::wstring &localeStr)
Sets the locale that will be used to read the data.
- void [SetType](#) (sparksee::gdb::type_t t)
Sets the type to be loaded.
- void [SetAttributes](#) (sparksee::gdb::AttributeList &attrs)
Sets the list of Attributes.
- void [SetAttributePositions](#) (sparksee::gdb::Int32List &attrsPos)
Sets the list of attribute positions.
- void [SetTimestampFormat](#) (const std::wstring ×tampFormat)
Sets a specific timestamp format.

- void [SetFrequency](#) ([sparksee::gdb::int32_t](#) freq)
Sets the frequency of listener notification.

Protected Types

- enum [Mode](#) {
 [ONE_PHASE](#),
 [TWO_PHASES](#),
 [N_PHASES](#) }
Load can work in different ways.

Protected Member Functions

- [sparksee::gdb::bool_t CanRun](#) ()
Checks that all the required settings are ready to run.
- void [Run](#) ([Mode](#) ph, [sparksee::gdb::int32_t](#) par) throw ([sparksee::gdb::IOException](#), [sparksee::gdb::Error](#))
Runs load process.
- void [NotifyListeners](#) (const [TypeLoaderEvent](#) &ev)
Notifies progress to all registered listeners.

5.19.1 Detailed Description

[EdgeTypeLoader](#) class.

Specific [TypeLoader](#) implementation for edge types.

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.19.2 Member Enumeration Documentation

5.19.2.1 enum [TypeLoader::Mode](#) [protected, inherited]

Load can work in different ways.

Enumerator:

- [ONE_PHASE](#)** Performs the load in a phases.
Load all objects an attributes at the same time.
- [TWO_PHASES](#)** Performs the load in two phases.
Firstly load all objects (and create them if necessary) and secondly loads all the attributes.
Working on this mode it is necessary to build a temporary file.

N_PHASES Performs the load in N phases.

Firstly load all objects (and create them if necessary) and secondly loads all the attributes. But in this case, attributes are loaded one by one. This way, if there are three attributes, then 4 traverses to the [RowReader](#) are necessary.

Working on this mode it is necessary to build a temporary file.

5.19.3 Constructor & Destructor Documentation

5.19.3.1 `EdgeTypeLoader::EdgeTypeLoader (RowReader & rr, sparksee::gdb::Graph & g, sparksee::gdb::type_t t, sparksee::gdb::AttributeList & attrs, sparksee::gdb::Int32List & attrPos, sparksee::gdb::int32_t hPos, sparksee::gdb::int32_t tPos, sparksee::gdb::attr_t hAttr, sparksee::gdb::attr_t tAttr) [inline]`

Creates a new instance.

Parameters:

rr [in] Input [RowReader](#).

g [in] [Graph](#).

t [in] [Type](#) identifier.

attrs [in] [Attribute](#) identifiers to be loaded.

attrPos [in] [Attribute](#) positions (column index >=0). to all listeners.

hPos [in] The position (index column) for the head value.

tPos [in] The position (index column) for the tail value.

hAttr [in] The attribute identifier for the head.

tAttr [in] The attribute identifier for the tail.

5.19.4 Member Function Documentation

5.19.4.1 `void EdgeTypeLoader::SetHeadAttribute (sparksee::gdb::attr_t attr) [inline]`

Sets the attribute that will be used to find the head of the edge.

This method is protected because only the Edge loaders should have it.

Parameters:

attr [in] Head [Attribute](#)

Reimplemented from [TypeLoader](#).

5.19.4.2 `void EdgeTypeLoader::SetHeadPosition (sparksee::gdb::int32_t pos) [inline]`

Sets the position of the head attribute in the source data.

This method is protected because only the Edge loaders should have it.

Parameters:

pos [in] Head position

Reimplemented from [TypeLoader](#).

5.19.4.3 void EdgeTypeLoader::SetTailAttribute (sparksee::gdb::attr_t *attr*) [inline]

Sets the attribute that will be used to find the tail of the edge.

This method is protected because only the Edge loaders should have it.

Parameters:

attr [in] Tail [Attribute](#)

Reimplemented from [TypeLoader](#).

5.19.4.4 void EdgeTypeLoader::SetTailPosition (sparksee::gdb::int32_t *pos*) [inline]

Sets the position of the tail attribute in the source data.

This method is protected because only the Edge loaders should have it.

Parameters:

pos [in] Tail position

Reimplemented from [TypeLoader](#).

5.19.4.5 sparksee::gdb::bool_t TypeLoader::CanRun () [protected, inherited]

Checks that all the required settings are ready to run.

Returns:

Returns true if all the settings are ready.

5.19.4.6 void TypeLoader::Run (Mode *ph*, sparksee::gdb::int32_t *par*) throw (sparksee::gdb::IOException, sparksee::gdb::Error) [protected, inherited]

Runs load process.

Exceptions:

[IOException](#) If bad things happen reading from the [RowReader](#).

Parameters:

ph [in] The load mode.

par [in] Number of horizontal partitions to perform the load.

5.19.4.7 void TypeLoader::NotifyListeners (const TypeLoaderEvent & *ev*) [protected, inherited]

Notifies progress to all registered listeners.

Parameters:

ev [in] Progress event to be notified.

5.19.4.8 void TypeLoader::SetLogError (const std::wstring & *path*) throw (sparksee::gdb::IOException) [inherited]

Sets a log error file.

By default errors are thrown as a exception and the load process ends. If a log file is set, errors are logged there and the load process does not stop.

Exceptions:

[*IOException*](#) If bad things happen opening the file.

5.19.4.9 void TypeLoader::SetLogOff () [inherited]

Turns off all the error reporting.

The log file will not be created and no exceptions for invalid data will be thrown. If you just want to turn off the logs, but abort at the first error what you should do is not call this method and not set a logError file.

5.19.4.10 void TypeLoader::Register (TypeLoaderListener & *tel*) [inherited]

Registers a new listener.

Parameters:

← *tel* [TypeLoaderListener](#) to be registered.

5.19.4.11 void TypeLoader::SetRowReader (RowReader & *rr*) [inherited]

Sets the input data source.

Parameters:

rr [in] Input [RowReader](#).

5.19.4.12 void TypeLoader::SetGraph (sparksee::gdb::Graph & *g*) [inherited]

Sets the graph where the data will be loaded.

Parameters:

g [in] [Graph](#).

5.19.4.13 void TypeLoader::SetLocale (const std::wstring & *localeStr*) [inherited]

Sets the locale that will be used to read the data.

It should match the locale used in the rowreader.

Parameters:

localeStr [in] The locale string for the read data. See [CSVReader](#).

5.19.4.14 void TypeLoader::SetType (sparksee::gdb::type_t *t*) [inherited]

Sets the type to be loaded.

Parameters:

t [in] [Type](#) identifier.

5.19.4.15 void TypeLoader::SetAttributes (sparksee::gdb::AttributeList & *attrs*) [inherited]

Sets the list of Attributes.

Parameters:

attrs [in] [Attribute](#) identifiers to be loaded

5.19.4.16 void TypeLoader::SetAttributePositions (sparksee::gdb::Int32List & *attrsPos*) [inherited]

Sets the list of attribute positions.

attrsPos [in] [Attribute](#) positions (column index >=0).

5.19.4.17 void TypeLoader::SetFrequency (sparksee::gdb::int32_t *freq*) [inherited]

Sets the frequency of listener notification.

freq [in] Frequency in number of rows managed to notify progress to all listeners

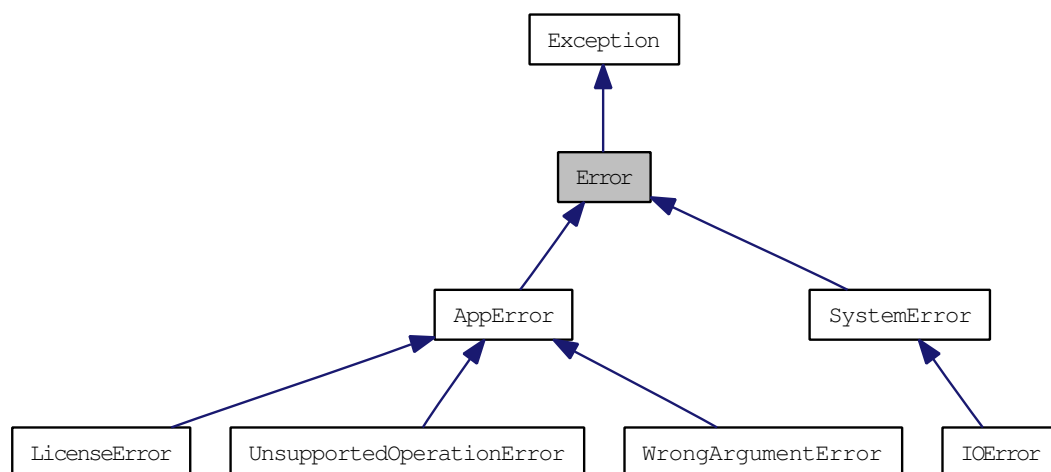
The documentation for this class was generated from the following file:

- [EdgeTypeLoader.h](#)

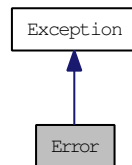
5.20 Error Class Reference

[Error](#) class.

Inheritance diagram for Error:



Collaboration diagram for Error:



Public Member Functions

- `Error ()`
Creates a new instance.
- `Error (const std::string &mess)`
Creates a new instance.
- `virtual ~Error ()`
Destructor.
- `const std::string & Message () const`
Gets the message of the exception.
- `void SetMessage (const std::string &mess)`
Sets the message of the exception.

Static Public Member Functions

- `static Error NewError (int32_t coreErrorCode)`
Creates a new `Error` instance from a `sparksee_core` error code.

Protected Attributes

- `std::string message`
Message of the exception.

5.20.1 Detailed Description

`Error` class.

An `Error` corresponds to an unexpected and unpredictable exception.

As all methods can throw an `Error` at any moment, it is not expected they declare the `Error` (or subclasses) they may throw.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.20.2 Constructor & Destructor Documentation

5.20.2.1 `Error::Error (const std::string & mess)`

Creates a new instance.

Parameters:

mess [in] Message of the exception.

5.20.3 Member Function Documentation

5.20.3.1 `static Error Error::NewError (int32_t coreErrorCode) [static]`

Creates a new [Error](#) instance from a sparksee_core error code.

Parameters:

coreErrorCode [in] Sparkseecore error code.

Returns:

Depending on the given sparksee_core error, this may return an [Error](#) instance or an specific [Error](#) subclass instance.

5.20.3.2 `const std::string& Exception::Message () const [inherited]`

Gets the message of the exception.

It should be called GetMessage but this is not possible because of a macro defined in Windows.

Returns:

The message of the exception.

5.20.3.3 `void Exception::SetMessage (const std::string & mess) [inherited]`

Sets the message of the exception.

Parameters:

mess [in] Message.

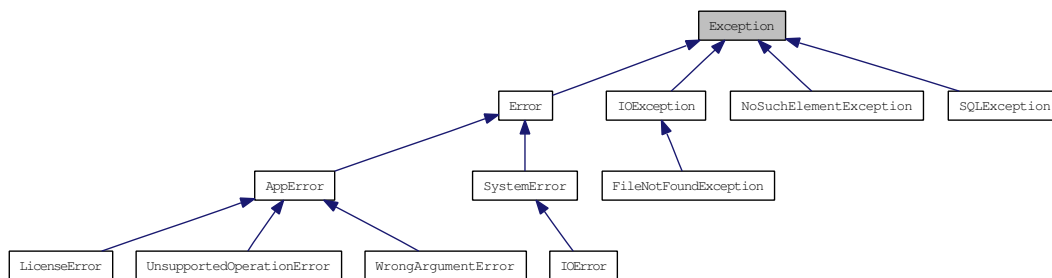
The documentation for this class was generated from the following file:

- Exception.h

5.21 Exception Class Reference

[Exception](#) class.

Inheritance diagram for Exception:



Public Member Functions

- [Exception](#) ()
Creates a new instance.
- [Exception](#) (const std::string &mess)
Creates a new instance.
- const std::string & [Message](#) () const
Gets the message of the exception.
- void [SetMessage](#) (const std::string &mess)
Sets the message of the exception.
- virtual [~Exception](#) ()
Destructor.

Protected Attributes

- std::string [message](#)
Message of the exception.

5.21.1 Detailed Description

[Exception](#) class.

This is the superclass of those exceptions that can be thrown during the normal execution of a program. It is expected all methods declare all [Exception](#) (or subclasses) they throw.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.21.2 Constructor & Destructor Documentation

5.21.2.1 Exception::Exception (const std::string & *mess*)

Creates a new instance.

Parameters:

mess [in] Message of the exception.

5.21.3 Member Function Documentation

5.21.3.1 const std::string& Exception::Message () const

Gets the message of the exception.

It should be called GetMessage but this is not possible because of a macro defined in Windows.

Returns:

The message of the exception.

5.21.3.2 void Exception::SetMessage (const std::string & *mess*)

Sets the message of the exception.

Parameters:

mess [in] Message.

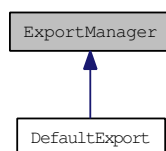
The documentation for this class was generated from the following file:

- Exception.h

5.22 ExportManager Class Reference

Defines how to export a graph to an external format.

Inheritance diagram for ExportManager:



Public Member Functions

- virtual [~ExportManager](#) ()
Destructor.
- virtual void [Prepare](#) ([Graph](#) *graph)=0

Prepares the graph for the export process.

- virtual void [Release](#) ()=0
Ends the export process.
- virtual [bool_t](#) [GetGraph](#) ([GraphExport](#) &graphExport)=0
Gets the graph export definition.
- virtual [bool_t](#) [GetNodeType](#) ([type_t](#) type, [NodeExport](#) &nodeExport)=0
Gets the default node export definition for the given node type.
- virtual [bool_t](#) [GetEdgeType](#) ([type_t](#) type, [EdgeExport](#) &edgeExport)=0
Gets the default node export definition for the given edge type.
- virtual [bool_t](#) [GetNode](#) ([oid_t](#) node, [NodeExport](#) &nodeExport)=0
Gets the node export definition for the given node.
- virtual [bool_t](#) [GetEdge](#) ([oid_t](#) edge, [EdgeExport](#) &edgeExport)=0
Gets the edge export definition for the given edge.
- virtual [bool_t](#) [EnableType](#) ([type_t](#) type)=0
Gets whether a node or edge type must be exported or not.

5.22.1 Detailed Description

Defines how to export a graph to an external format.

This is an interface which must be implemented by the user. While the export proces, a call for each node or edge type and node or edge object is done to get how to export that element.

It is possible to export a [Graph](#) to a diferent fortformats. Nowadays, available formats are defined in the [ExportType](#) enum.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.22.2 Member Function Documentation

5.22.2.1 virtual void ExportManager::Prepare ([Graph](#) * *graph*) [pure virtual]

Prepares the graph for the export process.

It is called once before the export process.

Parameters:

graph [Graph](#) to be exported.

Implemented in [DefaultExport](#).

5.22.2.2 virtual void ExportManager::Release () [pure virtual]

Ends the export process.

It is called once after the export process.

Implemented in [DefaultExport](#).

5.22.2.3 virtual bool_t ExportManager::GetGraph (GraphExport & *graphExport*) [pure virtual]

Gets the graph export definition.

Parameters:

graphExport [out] The [GraphExport](#) which defines how to export the graph.

Returns:

TRUE.

Implemented in [DefaultExport](#).

5.22.2.4 virtual bool_t ExportManager::GetNodeType (type_t *type*, NodeExport & *nodeExport*) [pure virtual]

Gets the default node export definition for the given node type.

GetNode has a higher priority than this function. That is, only if GetNode returns FALSE, the [NodeExport](#) of this function will be used.

Parameters:

type [in] Node type identifier.

nodeExport [out] The [NodeExport](#) which defines how to export the nodes of the given type.

Returns:

TRUE.

Implemented in [DefaultExport](#).

5.22.2.5 virtual bool_t ExportManager::GetEdgeType (type_t *type*, EdgeExport & *edgeExport*) [pure virtual]

Gets the default node export definition for the given edge type.

GetEdge has a higher priority than this function. That is, only if GetEdge returns FALSE, the [EdgeExport](#) of this function will be used.

Parameters:

type [in] Edge type identifier.

edgeExport [out] The [EdgeExport](#) which defines how to export the edges of the given type.

Returns:

TRUE.

Implemented in [DefaultExport](#).

5.22.2.6 `virtual bool_t ExportManager::GetNode (oid_t node, NodeExport & nodeExport)` [pure virtual]

Gets the node export definition for the given node.

Parameters:

node Node identifier.

nodeExport [out] The [NodeExport](#) which defines how to export given node.

Returns:

TRUE if the given [NodeExport](#) has been updated, otherwise FALSE will be returned and the default [NodeExport](#) for the type the node belongs to will be used.

Implemented in [DefaultExport](#).

5.22.2.7 `virtual bool_t ExportManager::GetEdge (oid_t edge, EdgeExport & edgeExport)` [pure virtual]

Gets the edge export definition for the given edge.

Parameters:

edge Edge identifier.

edgeExport [out] The [EdgeExport](#) which defines how to export given edge.

Returns:

TRUE if the given [EdgeExport](#) has been updated, otherwise FALSE will be returned and the default [EdgeExport](#) for the type the edge belongs to will be used.

Implemented in [DefaultExport](#).

5.22.2.8 `virtual bool_t ExportManager::EnableType (type_t type)` [pure virtual]

Gets whether a node or edge type must be exported or not.

Parameters:

type Node or edge type identifier.

Returns:

If TRUE all objects of the given type will be exported, otherwise they will not be exported.

Implemented in [DefaultExport](#).

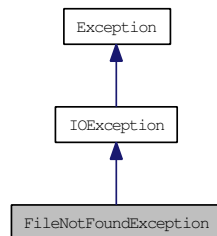
The documentation for this class was generated from the following file:

- [Export.h](#)

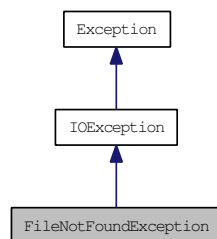
5.23 FileNotFoundException Class Reference

File not found exception class.

Inheritance diagram for FileNotFoundException:



Collaboration diagram for FileNotFoundException:



Public Member Functions

- [FileNotFoundException \(\)](#)
Creates a new instance.
- [FileNotFoundException \(const std::string &mess\)](#)
Creates a new instance.
- virtual [~FileNotFoundException \(\)](#)
Destructor.
- const std::string & [Message \(\)](#) const
Gets the message of the exception.
- void [SetMessage](#) (const std::string &mess)
Sets the message of the exception.

Protected Attributes

- std::string [message](#)
Message of the exception.

5.23.1 Detailed Description

File not found exception class.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.23.2 Constructor & Destructor Documentation

5.23.2.1 FileNotFoundException::FileNotFoundException (const std::string & mess)

Creates a new instance.

Parameters:

mess [in] Message of the exception.

5.23.3 Member Function Documentation

5.23.3.1 const std::string& Exception::Message () const [inherited]

Gets the message of the exception.

It should be called GetMessage but this is not possible because of a macro defined in Windows.

Returns:

The message of the exception.

5.23.3.2 void Exception::SetMessage (const std::string & mess) [inherited]

Sets the message of the exception.

Parameters:

mess [in] Message.

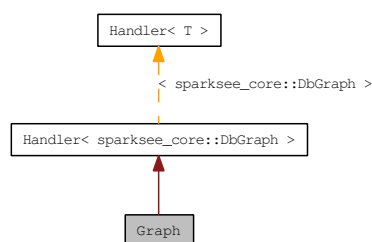
The documentation for this class was generated from the following file:

- Exception.h

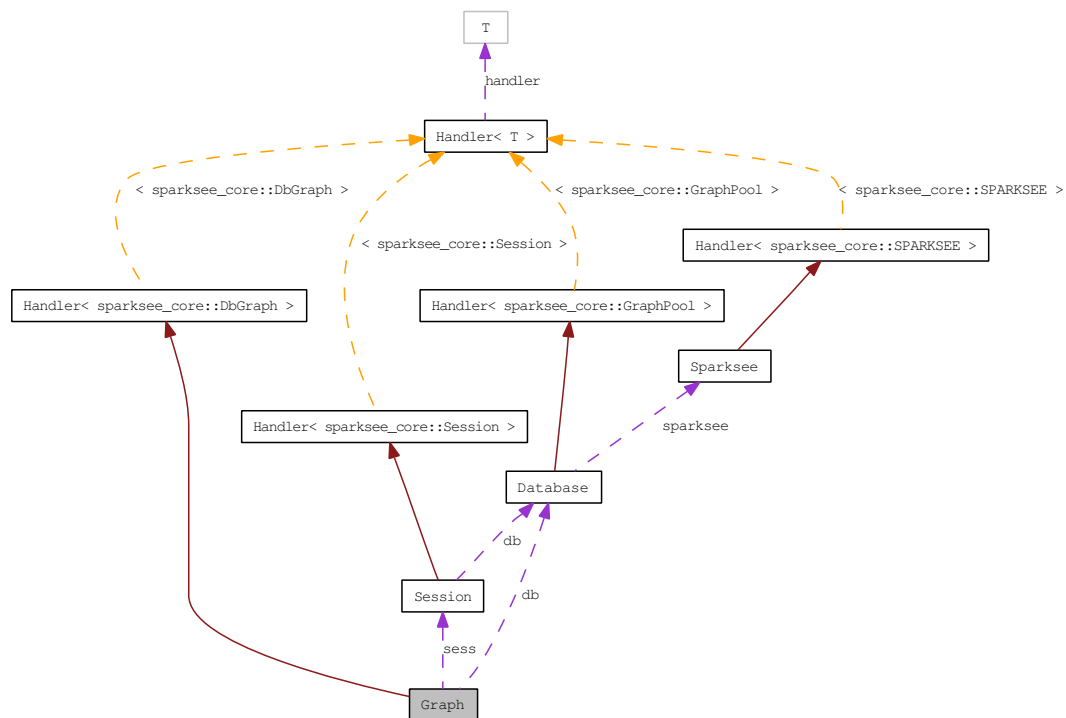
5.24 Graph Class Reference

[Graph](#) class.

Inheritance diagram for Graph:



Collaboration diagram for Graph:



Public Member Functions

- `virtual ~Graph ()`
Destructor.
- `type_t NewNodeType (const std::wstring &name)`
Creates a new node type.
- `oid_t NewNode (type_t type)`
Creates a new node instance.
- `type_t NewEdgeType (const std::wstring &name, bool_t directed, bool_t neighbors)`
Creates a new edge type.
- `type_t NewRestrictedEdgeType (const std::wstring &name, type_t tail, type_t head, bool_t neighbors)`
Creates a new restricted edge type.
- `oid_t NewEdge (type_t type, oid_t tail, oid_t head)`
Creates a new edge instance.
- `oid_t NewEdge (type_t type, attr_t tailAttr, Value &tailV, attr_t headAttr, Value &headV)`
Creates a new edge instance.
- `int64_t CountNodes ()`

Gets the number of nodes.

- `int64_t CountEdges ()`

Gets the number of edges.

- `EdgeData * GetEdgeData (oid_t edge)`

Gets information about an edge.

- `oid_t GetEdgePeer (oid_t edge, oid_t node)`

Gets the other end for the given edge.

- `void Drop (oid_t oid)`

Drops the given OID.

- `void Drop (Objects *objs)`

Drops all the OIDs from the given collection.

- `type_t GetObjectType (oid_t oid)`

*Gets the *Sparksee* node or edge type identifier for the given OID.*

- `attr_t NewAttribute (type_t type, const std::wstring &name, DataType dt, AttributeKind kind)`

Creates a new attribute.

- `attr_t NewAttribute (type_t type, const std::wstring &name, DataType dt, AttributeKind kind, Value &defaultValue)`

Creates a new attribute with a default value.

- `attr_t NewSessionAttribute (type_t type, DataType dt, AttributeKind kind)`

*Creates a new *Session* attribute.*

- `attr_t NewSessionAttribute (type_t type, DataType dt, AttributeKind kind, Value &defaultValue)`

*Creates a new *Session* attribute with a default value.*

- `void SetAttributeDefaultValue (attr_t attr, Value &v)`

Sets a default value for an attribute.

- `void IndexAttribute (attr_t attr, AttributeKind kind)`

Updates the index of the given attribute.

- `void GetAttribute (oid_t oid, attr_t attr, Value &v)`

*Gets the *Value* for the given attribute and OID.*

- `Value * GetAttribute (oid_t oid, attr_t attr)`

*Gets the *Value* for the given attribute and OID.*

- `TextStream * GetAttributeText (oid_t oid, attr_t attr)`

*Gets the read-only *TextStream* for the given text attribute and OID.*

- `void SetAttributeText (oid_t oid, attr_t attr, TextStream *tstream)`

*Sets the writable *TextStream* for the given text attribute and OID.*

- void **SetAttribute** (oid_t oid, attr_t attr, Value &v)
Sets the Value for the given attribute and OID.
- AttributeStatistics * **GetAttributeStatistics** (attr_t attr, bool_t basic)
Gets statistics from the given attribute.
- int64_t **GetAttributeIntervalCount** (attr_t attr, Value &lower, bool_t includeLower, Value &higher, bool_t includeHigher)
Gets how many objects have a value into the given range for the given attribute.
- type_t **FindType** (const std::wstring &name)
Gets the Sparksee type identifier for the given type name.
- Type * **GetType** (type_t type)
Gets information about the given type.
- void **RemoveType** (type_t type)
Removes the given type.
- void **RenameType** (const std::wstring &oldName, const std::wstring &newName)
Renames a type.
- void **RenameType** (type_t type, const std::wstring &newName)
Renames a type.
- attr_t **FindAttribute** (type_t type, const std::wstring &name)
Gets the Sparksee attribute identifier for the given type identifier and attribute name.
- Attribute * **GetAttribute** (attr_t attr)
Gets information about the given attribute.
- void **RemoveAttribute** (attr_t attr)
Removes the given attribute.
- void **RenameAttribute** (attr_t attr, const std::wstring &newName)
Renames an attribute.
- oid_t **FindObject** (attr_t attr, Value &v)
Finds one object having the given Value for the given attribute.
- oid_t **FindOrCreateObject** (attr_t attr, Value &v)
Finds one object having the given Value for the attribute or it creates one does not exist any.
- Objects * **Select** (type_t type)
Selects all OIDs belonging to the given type.
- Objects * **Select** (attr_t attr, Condition cond, const Value &v)
Selects all OIDs satisfying the given condition for the given attribute.

- **Objects * Select** (**attr_t** attr, **Condition** cond, const **Value** &lower, const **Value** &higher)
Selects all OIDs satisfying the given condition for the given attribute.
- **Objects * Select** (**attr_t** attr, **Condition** cond, const **Value** &v, const **Objects** *restriction)
Selects all OIDs satisfying the given condition for the given attribute.
- **Objects * Select** (**attr_t** attr, **Condition** cond, const **Value** &lower, const **Value** &higher, const **Objects** *restriction)
Selects all OIDs satisfying the given condition for the given attribute.
- **Objects * Explode** (**oid_t** oid, **type_t** etype, **EdgesDirection** dir)
Selects all edges from or to the given node OID and for the given edge type.
- **Objects * Explode** (**Objects** *objs, **type_t** etype, **EdgesDirection** dir)
Selects all edges from or to each of the node OID in the given collection and for the given edge type.
- **int64_t Degree** (**oid_t** oid, **type_t** etype, **EdgesDirection** dir)
Gets the number of edges from or to the given node OID and for the given edge type.
- **Objects * Neighbors** (**oid_t** oid, **type_t** etype, **EdgesDirection** dir)
Selects all neighbor nodes from or to the given node OID and for the given edge type.
- **Objects * Neighbors** (**Objects** *objs, **type_t** etype, **EdgesDirection** dir)
Selects all neighbor nodes from or to each of the node OID in the given collection and for the given edge type.
- **Objects * Edges** (**type_t** etype, **oid_t** tail, **oid_t** head)
Gets all the edges of the given type between two given nodes (tail and head).
- **oid_t FindEdge** (**type_t** etype, **oid_t** tail, **oid_t** head)
Gets any of the edges of the given type between two given nodes (tail and head).
- **oid_t FindOrCreateEdge** (**type_t** etype, **oid_t** tail, **oid_t** head)
Gets any of the edges of the specified type between two given nodes (tail and head).
- **Objects * Tails** (**Objects** *edges)
Gets all the tails from the given edges collection.
- **Objects * Heads** (**Objects** *edges)
Gets all the heads from the given edges collection.
- **void TailsAndHeads** (**Objects** *edges, **Objects** *tails, **Objects** *heads)
Gets all the tails and heads from the given edges collection.
- **TypeList * FindNodeTypes** ()
*Gets all existing *Sparksee* node type identifiers.*
- **TypeList * FindEdgeTypes** ()
*Gets all existing *Sparksee* edge type identifiers.*

- [TypeList](#) * [FindTypes](#) ()
Gets all existing [Sparksee](#) node and edge type identifiers.
- [AttributeList](#) * [FindAttributes](#) ([type_t](#) type)
Gets all existing [Sparksee](#) attribute identifiers for the given type identifier.
- [AttributeList](#) * [GetAttributes](#) ([oid_t](#) oid)
Gets all [Sparksee](#) attribute identifiers with a non-NULL value for the given [Sparksee](#) OID.
- [Values](#) * [GetValues](#) ([attr_t](#) attr)
Gets the [Value](#) collection for the given attribute.
- void [DumpData](#) (const std::wstring &file) throw (sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)
Dumps logical data to a file.
- void [DumpStorage](#) (const std::wstring &file) throw (sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)
Dumps internal storage data to a file.
- void [Export](#) (const std::wstring &file, [ExportType](#) type, [ExportManager](#) *e) throw (sparksee::gdb::IOException)
Exports the [Graph](#).
- void [Backup](#) (const std::wstring &file) throw (sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)
Dumps all the data to a backup file.

Friends

- class [Session](#)
- class [Values](#)
- class [ValuesIterator](#)

5.24.1 Detailed Description

[Graph](#) class.

Each [Database](#) has a [Graph](#) associated, which is the persistent graph which contains all data stored into the graph database and is retrieved from a [Session](#).

Check out the 'API' and the 'SPARKSEE graph database' sections in the SPARKSEE User Manual for more details on the use of this class.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.24.2 Member Function Documentation

5.24.2.1 `type_t Graph::NewNodeType (const std::wstring & name)`

Creates a new node type.

Parameters:

name [in] Unique name for the new node type.

Returns:

Unique [Sparksee](#) type identifier.

5.24.2.2 `oid_t Graph::NewNode (type_t type)`

Creates a new node instance.

Parameters:

type [in] [Sparksee](#) type identifier.

Returns:

Unique OID of the new node instance.

5.24.2.3 `type_t Graph::NewEdgeType (const std::wstring & name, bool_t directed, bool_t neighbors)`

Creates a new edge type.

Parameters:

name [in] Unique name for the new edge type.

directed [in] If TRUE, this creates a directed edge type, otherwise this creates an undirected edge type.

neighbors [in] If TRUE, this indexes neighbor nodes, otherwise not.

Returns:

Unique [Sparksee](#) type identifier.

5.24.2.4 `type_t Graph::NewRestrictedEdgeType (const std::wstring & name, type_t tail, type_t head, bool_t neighbors)`

Creates a new restricted edge type.

Parameters:

name [in] Unique name for the new edge type.

tail [in] Tail [Sparksee](#) node type identifier.

head [in] Head [Sparksee](#) node type identifier.

neighbors [in] If TRUE, this indexes neighbor nodes, otherwise not.

Returns:

Unique [Sparksee](#) type identifier.

5.24.2.5 `oid_t Graph::NewEdge (type_t type, oid_t tail, oid_t head)`

Creates a new edge instance.

Parameters:

type [in] [Sparksee](#) type identifier.
tail [in] Source [Sparksee](#) OID.
head [in] Target [Sparksee](#) OID.

Returns:

Unique OID of the new edge instance.

5.24.2.6 `oid_t Graph::NewEdge (type_t type, attr_t tailAttr, Value & tailV, attr_t headAttr, Value & headV)`

Creates a new edge instance.

The tail of the edge will be any node having the given tailV [Value](#) for the given tailAttr attribute identifier, and the head of the edge will be any node having the given headV [Value](#) for the given headAttr attribute identifier.

Parameters:

type [in] [Sparksee](#) type identifier.
tailAttr [in] [Sparksee](#) attribute identifier.
tailV [in] [Value](#).
headAttr [in] [Sparksee](#) attribute identifier.
headV [in] [Value](#).

Returns:

Unique OID of the new edge instance.

5.24.2.7 `int64_t Graph::CountNodes ()`

Gets the number of nodes.

Returns:

The number of nodes.

5.24.2.8 `int64_t Graph::CountEdges ()`

Gets the number of edges.

Returns:

The number of edges.

5.24.2.9 `EdgeData* Graph::GetEdgeData (oid_t edge)`

Gets information about an edge.

Parameters:

edge [in] [Sparksee](#) edge identifier.

Returns:

An [EdgeData](#) instance.

5.24.2.10 `oid_t Graph::GetEdgePeer (oid_t edge, oid_t node)`

Gets the other end for the given edge.

Parameters:

edge [in] [Sparksee](#) edge identifier.

node [in] [Sparksee](#) node identifier. It must be one of the ends of the edge.

Returns:

The other end of the edge.

5.24.2.11 `void Graph::Drop (oid_t oid)`

Drops the given OID.

It also removes its edges as well as its attribute values.

Parameters:

oid [in] [Sparksee](#) OID to be removed.

5.24.2.12 `void Graph::Drop (Objects * objs)`

Drops all the OIDs from the given collection.

See Drop method with the single OID parameter. This performs that call for all the elements into the collection.

Parameters:

objs [in] [Objects](#) collection with the OIDs to be removed.

5.24.2.13 `type_t Graph::GetObjectType (oid_t oid)`

Gets the [Sparksee](#) node or edge type identifier for the given OID.

Parameters:

oid [in] [Sparksee](#) OID.

Returns:

[Sparksee](#) node or edge type identifier.

5.24.2.14 attr_t Graph::NewAttribute (type_t *type*, const std::wstring & *name*, DataType *dt*, AttributeKind *kind*)

Creates a new attribute.

Parameters:

type [in] [Sparksee](#) node or edge type identifier.
name [in] Unique name for the new attribute.
dt [in] Data type for the new attribute.
kind [in] [Attribute](#) kind.

Returns:

Unique [Sparksee](#) attribute identifier.

5.24.2.15 attr_t Graph::NewAttribute (type_t *type*, const std::wstring & *name*, DataType *dt*, AttributeKind *kind*, Value & *defaultValue*)

Creates a new attribute with a default value.

Parameters:

type [in] [Sparksee](#) node or edge type identifier.
name [in] Unique name for the new attribute.
dt [in] Data type for the new attribute.
kind [in] [Attribute](#) kind.
defaultValue [in] The default value to use in each new node/edge.

Returns:

Unique [Sparksee](#) attribute identifier.

5.24.2.16 attr_t Graph::NewSessionAttribute (type_t *type*, DataType *dt*, AttributeKind *kind*)

Creates a new [Session](#) attribute.

[Session](#) attributes are exclusive for the [Session](#) (just its [Session](#) can use the attribute) and are automatically removed when the [Session](#) is closed (thus, attribute data is not persistent into the database).

Since they are not persistent, they cannot be retrieved from the database, so they do not have an identifier name.

Parameters:

type [in] [Sparksee](#) node or edge type identifier.
dt [in] Data type for the new attribute.
kind [in] [Attribute](#) kind.

Returns:

Unique [Sparksee](#) attribute identifier.

5.24.2.17 attr_t Graph::NewSessionAttribute (type_t *type*, DataType *dt*, AttributeKind *kind*, Value & *defaultValue*)

Creates a new [Session](#) attribute with a default value.

[Session](#) attributes are exclusive for the [Session](#) (just its [Session](#) can use the attribute) and are automatically removed when the [Session](#) is closed (thus, attribute data is not persistent into the database).

Since they are not persistent, they cannot be retrieved from the database, so they do not have an identifier name.

Parameters:

- type* [in] [Sparksee](#) node or edge type identifier.
- dt* [in] Data type for the new attribute.
- kind* [in] [Attribute](#) kind.
- defaultValue* [in] The default value to use in each new node/edge.

Returns:

Unique [Sparksee](#) attribute identifier.

5.24.2.18 void Graph::SetAttributeDefaultValue (attr_t *attr*, Value & *v*)

Sets a default value for an attribute.

The default value will be applied to all the new nodes or edges.

The given value must have the same DataType as the attribute or be a NULL value to remove the current default value.

Parameters:

- attr* [in] The attribute.
- v* [in] The default value to use for this attribute.

5.24.2.19 void Graph::IndexAttribute (attr_t *attr*, AttributeKind *kind*)

Updates the index of the given attribute.

This just works if the current index of the attribute corresponds to the AttributeKind Basic and the new one is Indexed or Unique.

Parameters:

- attr* [in] [Sparksee](#) attribute identifier.
- kind* [in] [Attribute](#) kind.

5.24.2.20 void Graph::GetAttribute (oid_t *oid*, attr_t *attr*, Value & *v*)

Gets the [Value](#) for the given attribute and OID.

Parameters:

- oid* [in] [Sparksee](#) OID.
- attr* [in] [Sparksee](#) attribute identifier.
- v* [in|out] [Value](#) for the given attribute and for the given OID.

5.24.2.21 Value* Graph::GetAttribute (oid_t oid, attr_t attr)

Gets the [Value](#) for the given attribute and OID.

The other version of this call, where the [Value](#) is an output parameter instead of the return, is better because it allows the user to reuse an existing [Value](#) instance, whereas this call always creates a new [Value](#) instance to be returned.

It never returns NULL. Thus, in case the OID has a NULL value for the attribute it returns a NULL [Value](#) instance.

Parameters:

oid [in] [Sparksee](#) OID.

attr [in] [Sparksee](#) attribute identifier.

Returns:

A new [Value](#) instance having the attribute value for the given OID.

5.24.2.22 TextStream* Graph::GetAttributeText (oid_t oid, attr_t attr)

Gets the read-only [TextStream](#) for the given text attribute and OID.

Parameters:

oid [in] [Sparksee](#) OID.

attr [in] [Sparksee](#) attribute identifier.

Returns:

A [TextStream](#). This returns a [TextStream](#) to read.

5.24.2.23 void Graph::SetAttributeText (oid_t oid, attr_t attr, TextStream * tstream)

Sets the writable [TextStream](#) for the given text attribute and OID.

Parameters:

oid [in] [Sparksee](#) OID.

attr [in] [Sparksee](#) attribute identifier.

tstream [in] New Text value. This corresponds to a [TextStream](#) to write.

5.24.2.24 void Graph::SetAttribute (oid_t oid, attr_t attr, Value & v)

Sets the [Value](#) for the given attribute and OID.

Parameters:

oid [in] [Sparksee](#) OID.

attr [in] [Sparksee](#) attribute identifier.

v [in] [Value](#) for the given attribute and for the given OID.

5.24.2.25 `AttributeStatistics* Graph::GetAttributeStatistics (attr_t attr, bool_t basic)`

Gets statistics from the given attribute.

Parameters:

attr [in] [Sparksee](#) attribute identifier.

basic [in] If FALSE all statistics are computed, if TRUE just those statistics marked as basic will be computed (see description of the [AttributeStatistics](#) class). Of course, computing just basic statistics will be faster than computing all of them.

Returns:

An [AttributeStatistics](#) instance.

5.24.2.26 `int64_t Graph::GetAttributeIntervalCount (attr_t attr, Value & lower, bool_t includeLower, Value & higher, bool_t includeHigher)`

Gets how many objects have a value into the given range for the given attribute.

This only works for the attributes with the AttributeKind Indexed or Unique.

Given values must belong to the same DataType than the attribute.

Parameters:

attr [in] [Sparksee](#) attribute identifier.

lower [in] Lower bound [Value](#) of the range.

includeLower [in] If TRUE, include lower bound [Value](#) of the range.

higher [in] Higher bound [Value](#) of the range.

includeHigher [in] If TRUE, include higher bound [Value](#) of the range.

Returns:

Number of objects having a value into the given range.

5.24.2.27 `type_t Graph::FindType (const std::wstring & name)`

Gets the [Sparksee](#) type identifier for the given type name.

Parameters:

name [in] Unique type name.

Returns:

The [Sparksee](#) type identifier for the given type name or the [Type](#) InvalidType if there is no type with the given name.

5.24.2.28 `Type* Graph::GetType (type_t type)`

Gets information about the given type.

Parameters:

type [in] [Sparksee](#) type identifier.

Returns:

The [Type](#) for the given [Sparksee](#) type identifier.

5.24.2.29 void Graph::RemoveType (type_t type)

Removes the given type.

This fails if there exist attributes defined for the type or if there exist restricted edges referencing this type.

Parameters:

type [in] [Sparksee](#) type identifier.

5.24.2.30 void Graph::RenameType (const std::wstring & oldName, const std::wstring & newName)

Renames a type.

The new name must be available.

Parameters:

oldName [in] The current name of the type to be renamed.

newName [in] The new name for the type.

5.24.2.31 void Graph::RenameType (type_t type, const std::wstring & newName)

Renames a type.

The new name must be available.

Parameters:

type [in] The type to be renamed.

newName [in] The new name for the type.

5.24.2.32 attr_t Graph::FindAttribute (type_t type, const std::wstring & name)

Gets the [Sparksee](#) attribute identifier for the given type identifier and attribute name.

Parameters:

type [in] [Sparksee](#) type identifier.

name [in] Unique attribute name.

Returns:

The [Sparksee](#) attribute identifier for the given type and attribute name or InvalidAttribute if there is no attribute with the given name for the given type.

5.24.2.33 `Attribute* Graph::GetAttribute (attr_t attr)`

Gets information about the given attribute.

Parameters:

attr [in] [Sparksee](#) attribute identifier.

Returns:

The [Attribute](#) for the given [Sparksee](#) attribute identifier.

5.24.2.34 `void Graph::RemoveAttribute (attr_t attr)`

Removes the given attribute.

Parameters:

attr [in] [Sparksee](#) attribute identifier.

5.24.2.35 `void Graph::RenameAttribute (attr_t attr, const std::wstring & newName)`

Renames an attribute.

The new name must be available.

Parameters:

attr [in] [Sparksee](#) attribute identifier.

newName [in] The new name for the attribute.

5.24.2.36 `oid_t Graph::FindObject (attr_t attr, Value & v)`

Finds one object having the given [Value](#) for the given attribute.

If there are more than one, then any of them will be returned. And in case there are no object having this [Value](#), the [Objects](#) InvalidOID will be returned.

Parameters:

attr [in] [Sparksee](#) attribute identifier.

v [in] [Value](#).

Returns:

[Sparksee](#) OID or the [Objects](#) InvalidOID.

5.24.2.37 `oid_t Graph::FindOrCreateObject (attr_t attr, Value & v)`

Finds one object having the given [Value](#) for the attribute or it creates one does not exist any.

If the attribute is a node or edge attribute and at least one node/edge with that value is found, it returns one of them. But if it does not exist, then: If it's a node attribute it will create it and set the attribute. If it's an edge attribute it will return the InvalidOID.

Using this method with a global attribute will return the InvalidOID.

Parameters:

attr [in] [Sparksee](#) attribute identifier.
v [in] [Value](#).

Returns:

[Sparksee](#) OID or the [Objects](#) InvalidOID.

5.24.2.38 [Objects](#)* [Graph](#)::Select (type_t *type*)

Selects all OIDs belonging to the given type.

Parameters:

type [in] [Sparksee](#) type identifier.

Returns:

[Objects](#) instance.

5.24.2.39 [Objects](#)* [Graph](#)::Select (attr_t *attr*, Condition *cond*, const Value & *v*)

Selects all OIDs satisfying the given condition for the given attribute.

Parameters:

attr [in] [Sparksee](#) attribute identifier.
cond [in] Condition to be satisfied.
v [in] [Value](#) to be satisfied.

Returns:

[Objects](#) instance.

5.24.2.40 [Objects](#)* [Graph](#)::Select (attr_t *attr*, Condition *cond*, const Value & *lower*, const Value & *higher*)

Selects all OIDs satisfying the given condition for the given attribute.

This allows to perform the Between operation, thus it has two [Value](#) arguments.

Parameters:

attr [in] [Sparksee](#) attribute identifier.
cond [in] Condition to be satisfied. It must be the Between Condition.
lower [in] Lower-bound [Value](#) to be satisfied.
higher [in] Higher-bound [Value](#) to be satisfied.

Returns:

[Objects](#) instance.

5.24.2.41 Objects* Graph::Select (attr_t *attr*, Condition *cond*, const Value & *v*, const Objects * *restriction*)

Selects all OIDs satisfying the given condition for the given attribute.

Parameters:

attr [in] [Sparksee](#) attribute identifier.

cond [in] Condition to be satisfied.

v [in] [Value](#) to be satisfied.

restriction [in] [Objects](#) to limit the select in this set of objects.

Returns:

[Objects](#) instance.

5.24.2.42 Objects* Graph::Select (attr_t *attr*, Condition *cond*, const Value & *lower*, const Value & *higher*, const Objects * *restriction*)

Selects all OIDs satisfying the given condition for the given attribute.

This allows to perform the Between operation, thus it has two [Value](#) arguments.

Parameters:

attr [in] [Sparksee](#) attribute identifier.

cond [in] Condition to be satisfied. It must be the Between Condition.

lower [in] Lower-bound [Value](#) to be satisfied.

higher [in] Higher-bound [Value](#) to be satisfied.

restriction [in] [Objects](#) to limit the select in this set of objects.

Returns:

[Objects](#) instance.

5.24.2.43 Objects* Graph::Explode (oid_t *oid*, type_t *etype*, EdgesDirection *dir*)

Selects all edges from or to the given node OID and for the given edge type.

Parameters:

oid [in] [Sparksee](#) node OID.

etype [in] [Sparksee](#) edge type identifier.

dir [in] Direction.

Returns:

[Objects](#) instance.

5.24.2.44 Objects* Graph::Explode (Objects * *objs*, type_t *etype*, EdgesDirection *dir*)

Selects all edges from or to each of the node OID in the given collection and for the given edge type.

Parameters:

objs [in] [Sparksee](#) node OID collection.
etype [in] [Sparksee](#) edge type identifier.
dir [in] Direction.

Returns:

[Objects](#) instance.

5.24.2.45 int64_t Graph::Degree (oid_t *oid*, type_t *etype*, EdgesDirection *dir*)

Gets the number of edges from or to the given node OID and for the given edge type.

Parameters:

oid [in] [Sparksee](#) node OID.
etype [in] [Sparksee](#) edge type identifier.
dir [in] Direction.

Returns:

The number of edges.

5.24.2.46 Objects* Graph::Neighbors (oid_t *oid*, type_t *etype*, EdgesDirection *dir*)

Selects all neighbor nodes from or to the given node OID and for the given edge type.

Parameters:

oid [in] [Sparksee](#) node OID.
etype [in] [Sparksee](#) edge type identifier.
dir [in] Direction.

Returns:

[Objects](#) instance.

5.24.2.47 Objects* Graph::Neighbors (Objects * *objs*, type_t *etype*, EdgesDirection *dir*)

Selects all neighbor nodes from or to each of the node OID in the given collection and for the given edge type.

Parameters:

objs [in] [Sparksee](#) node OID collection.
etype [in] [Sparksee](#) edge type identifier.
dir [in] Direction.

Returns:

[Objects](#) instance.

5.24.2.48 Objects* Graph::Edges (type_t etype, oid_t tail, oid_t head)

Gets all the edges of the given type between two given nodes (tail and head).

Parameters:

etype [in] [Sparksee](#) edge type identifier.

tail [in] Tail node identifier.

head [in] Head node identifier.

Returns:

[Objects](#) instance.

5.24.2.49 oid_t Graph::FindEdge (type_t etype, oid_t tail, oid_t head)

Gets any of the edges of the given type between two given nodes (tail and head).

If there are more than one, then any of them will be returned. And in case there are no edge between the given tail and head, the [Objects](#) InvalidOID will be returned.

Parameters:

etype [in] [Sparksee](#) edge type identifier.

tail [in] Tail node identifier.

head [in] Head node identifier.

Returns:

Any of the edges or the [Objects](#) InvalidOID.

5.24.2.50 oid_t Graph::FindOrCreateEdge (type_t etype, oid_t tail, oid_t head)

Gets any of the edges of the specified type between two given nodes (tail and head).

If it can not find any edge of this type between them it tries to create a new one.

Parameters:

etype [in] [Sparksee](#) edge type identifier.

tail [in] Tail node identifier.

head [in] Head node identifier.

Returns:

Any of the edges or the [Objects](#) InvalidOID.

5.24.2.51 Objects* Graph::Tails (Objects * edges)

Gets all the tails from the given edges collection.

Parameters:

edges [in] [Sparksee](#) edge identifier collection.

Returns:

The tails collection.

5.24.2.52 Objects* Graph::Heads (Objects * *edges*)

Gets all the heads from the given edges collection.

Parameters:

edges [in] [Sparksee](#) edge identifier collection.

Returns:

The heads collection.

5.24.2.53 void Graph::TailsAndHeads (Objects * *edges*, Objects * *tails*, Objects * *heads*)

Gets all the tails and heads from the given edges collection.

Parameters:

edges [in] [Sparksee](#) edge identifier collection.

tails [in|out] If not NULL, all the tails will be stored here.

heads [in|out] If not NULL, all the heads will be stored here.

5.24.2.54 TypeList* Graph::FindNodeTypes ()

Gets all existing [Sparksee](#) node type identifiers.

Returns:

[Sparksee](#) node type identifier list.

5.24.2.55 TypeList* Graph::FindEdgeTypes ()

Gets all existing [Sparksee](#) edge type identifiers.

Returns:

[Sparksee](#) edge type identifier list.

5.24.2.56 TypeList* Graph::FindTypes ()

Gets all existing [Sparksee](#) node and edge type identifiers.

Returns:

[Sparksee](#) node and edge type identifier list.

5.24.2.57 AttributeList* Graph::FindAttributes (type_t *type*)

Gets all existing [Sparksee](#) attribute identifiers for the given type identifier.

Parameters:

type [in] [Sparksee](#) type identifier.

Returns:

[Sparksee](#) attribute identifier list.

5.24.2.58 `AttributeList* Graph::GetAttributes (oid_t oid)`

Gets all [Sparksee](#) attribute identifiers with a non-NULL value for the given [Sparksee](#) OID.

Parameters:

oid [in] [Sparksee](#) OID.

Returns:

[Sparksee](#) attribute identifier list.

5.24.2.59 `Values* Graph::GetValues (attr_t attr)`

Gets the [Value](#) collection for the given attribute.

Parameters:

attr [in] [Sparksee](#) attribute identifier.

5.24.2.60 `void Graph::DumpData (const std::wstring & file) throw (sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)`

Dumps logical data to a file.

Parameters:

file [in] Output file path.

Exceptions:

[FileNotFoundException](#) If the given file cannot be created.

5.24.2.61 `void Graph::DumpStorage (const std::wstring & file) throw (sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)`

Dumps internal storage data to a file.

Parameters:

file [in] Output file path.

Exceptions:

[FileNotFoundException](#) If the given file cannot be created.

5.24.2.62 `void Graph::Export (const std::wstring & file, ExportType type, ExportManager * e) throw (sparksee::gdb::IOException)`

Exports the [Graph](#).

Parameters:

file [in] Output file.

type [in] Export type.

e [in] Defines how to do the export for each graph object.

5.24.2.63 `void Graph::Backup (const std::wstring & file) throw (sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)`

Dumps all the data to a backup file.

See the [Sparksee](#) class Restore method.

Parameters:

file [in] Output backup file path.

Exceptions:

[FileNotFoundException](#) If the given file cannot be created.

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

- Graph.h

5.25 GraphExport Class Reference

Stores the graph exporting values.

Public Member Functions

- [GraphExport](#) ()
Creates a new [GraphExport](#) instance.
- virtual [~GraphExport](#) ()
Destructor.
- void [SetDefaults](#) ()
Sets to default values.
- const std::wstring & [GetLabel](#) () const
Gets the graph label.
- void [SetLabel](#) (const std::wstring &l)
Sets the graph label.

5.25.1 Detailed Description

Stores the graph exporting values.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.25.2 Member Function Documentation

5.25.2.1 const std::wstring& GraphExport::GetLabel () const [inline]

Gets the graph label.

Returns:

The graph label.

5.25.2.2 void GraphExport::SetLabel (const std::wstring & l) [inline]

Sets the graph label.

Parameters:

l [in] The graph label.

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

- Export.h

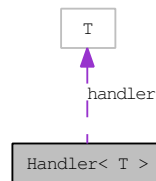
5.26 Handler< T > Class Template Reference

Handles a reference.

Inheritance diagram for Handler< T >:



Collaboration diagram for Handler< T >:



Public Member Functions

- [Handler](#) ()
Creates a new instance.
- [Handler](#) (T *h)
Creates a new instance with the given reference.
- virtual [~Handler](#) ()
Destructor.
- T * [GetHandler](#) ()
Gets the handled reference.
- const T * [GetHandler](#) () const
Gets the handled reference.

Protected Member Functions

- void [SetHandler](#) (T *h)
Sets the handled reference.
- void [FreeHandler](#) ()
Frees (deletes) the handled reference.
- [bool_t IsNull](#) () const
Gets if the handler is NULL.

5.26.1 Detailed Description

template<typename T> class Handler< T >

Handles a reference.

The handled reference is automatically destroyed (deleted) when the instance is destroyed.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

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

- `Handler.h`

5.27 Int32List Class Reference

[Sparksee](#) 32-bit signed integer list.

Public Member Functions

- `int32_t Count () const`
Number of elements in the list.
- `Int32ListIterator * Iterator ()`
Gets a new [Int32ListIterator](#).
- `Int32List ()`
Constructor.
- `Int32List (const std::vector< int32_t > &v)`
Constructor.
- `~Int32List ()`
Destructor.
- `void Add (int32_t value)`
Adds an 32-bit signed integer at the end of the list.
- `void Clear ()`
Clears the list.

5.27.1 Detailed Description

[Sparksee](#) 32-bit signed integer list.

It stores a 32-bit signed integer list.

Use [Int32ListIterator](#) to access all elements into this collection.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.27.2 Constructor & Destructor Documentation

5.27.2.1 Int32List::Int32List ()

Constructor.

This creates an empty list.

5.27.2.2 Int32List::Int32List (const std::vector< int32_t > & v)

Constructor.

Parameters:

v [in] Vector.

5.27.3 Member Function Documentation

5.27.3.1 int32_t Int32List::Count () const [inline]

Number of elements in the list.

Returns:

Number of elements in the list.

5.27.3.2 Int32ListIterator* Int32List::Iterator ()

Gets a new [Int32ListIterator](#).

Returns:

[Int32ListIterator](#) instance.

5.27.3.3 void Int32List::Add (int32_t value) [inline]

Adds an 32-bit signed integer at the end of the list.

Parameters:

type [in] The integer.

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

- Graph_data.h

5.28 Int32ListIterator Class Reference

[Int32List](#) iterator class.

Public Member Functions

- [~Int32ListIterator \(\)](#)

Destructor.

- [int32_t Next \(\)](#)

Moves to the next element.

- [bool_t HasNext \(\)](#)

Gets if there are more elements.

Friends

- class [Int32List](#)

5.28.1 Detailed Description

[Int32List](#) iterator class.

Iterator to traverse all the integer into a [Int32List](#) instance.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.28.2 Member Function Documentation

5.28.2.1 `int32_t Int32ListIterator::Next ()` [inline]

Moves to the next element.

Returns:

The next element.

5.28.2.2 `bool_t Int32ListIterator::HasNext ()` [inline]

Gets if there are more elements.

Returns:

TRUE if there are more elements, FALSE otherwise.

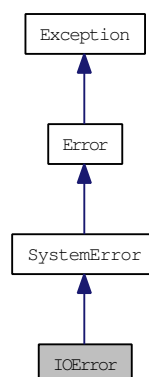
The documentation for this class was generated from the following file:

- `Graph_data.h`

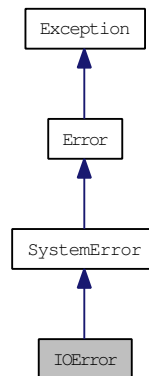
5.29 IOError Class Reference

IO error class.

Inheritance diagram for IOError:



Collaboration diagram for IOError:



Public Member Functions

- [IOError](#) ()
Creates a new instance.
- [IOError](#) (const std::string &mess)
Creates a new instance.
- virtual [~IOError](#) ()
Destructor.
- const std::string & [Message](#) () const
Gets the message of the exception.
- void [SetMessage](#) (const std::string &mess)
Sets the message of the exception.

Static Public Member Functions

- static [Error NewError](#) (int32_t coreErrorCode)
Creates a new [Error](#) instance from a sparksee_core error code.

Protected Attributes

- std::string [message](#)
Message of the exception.

5.29.1 Detailed Description

IO error class.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.29.2 Constructor & Destructor Documentation**5.29.2.1 IOError::IOError (const std::string & *mess*)**

Creates a new instance.

Parameters:

mess [in] Message of the exception.

5.29.3 Member Function Documentation**5.29.3.1 static Error Error::NewError (int32_t *coreErrorCode*) [static, inherited]**

Creates a new [Error](#) instance from a sparksee_core error code.

Parameters:

coreErrorCode [in] Sparkseecore error code.

Returns:

Depending on the given sparksee_core error, this may return an [Error](#) instance or an specific [Error](#) subclass instance.

5.29.3.2 const std::string& Exception::Message () const [inherited]

Gets the message of the exception.

It should be called GetMessage but this is not possible because of a macro defined in Windows.

Returns:

The message of the exception.

5.29.3.3 void Exception::SetMessage (const std::string & *mess*) [inherited]

Sets the message of the exception.

Parameters:

mess [in] Message.

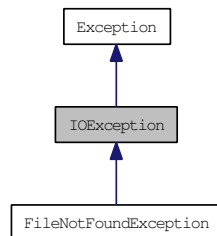
The documentation for this class was generated from the following file:

- Exception.h

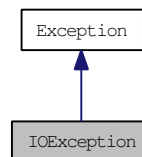
5.30 IOException Class Reference

IO exception class.

Inheritance diagram for IOException:



Collaboration diagram for IOException:



Public Member Functions

- [IOException](#) ()
Creates a new instance.
- [IOException](#) (const std::string &mess)
Creates a new instance.
- virtual [~IOException](#) ()
Destructor.
- const std::string & [Message](#) () const
Gets the message of the exception.
- void [SetMessage](#) (const std::string &mess)
Sets the message of the exception.

Protected Attributes

- std::string [message](#)
Message of the exception.

5.30.1 Detailed Description

IO exception class.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.30.2 Constructor & Destructor Documentation

5.30.2.1 IOException::IOException (const std::string & *mess*)

Creates a new instance.

Parameters:

mess [in] Message of the exception.

5.30.3 Member Function Documentation

5.30.3.1 const std::string& Exception::Message () const [inherited]

Gets the message of the exception.

It should be called GetMessage but this is not possible because of a macro defined in Windows.

Returns:

The message of the exception.

5.30.3.2 void Exception::SetMessage (const std::string & *mess*) [inherited]

Sets the message of the exception.

Parameters:

mess [in] Message.

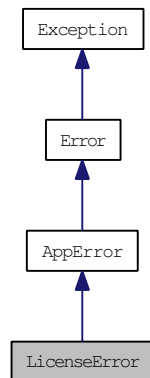
The documentation for this class was generated from the following file:

- Exception.h

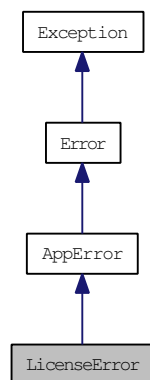
5.31 LicenseError Class Reference

License error class.

Inheritance diagram for LicenseError:



Collaboration diagram for LicenseError:



Public Member Functions

- [LicenseError](#) ()
Creates a new instance.
- [LicenseError](#) (const std::string &mess)
Creates a new instance.
- virtual [~LicenseError](#) ()
Destructor.
- const std::string & [Message](#) () const
Gets the message of the exception.
- void [SetMessage](#) (const std::string &mess)
Sets the message of the exception.

Static Public Member Functions

- static [Error](#) [NewError](#) ([int32_t](#) *coreErrorCode*)
Creates a new [Error](#) instance from a sparksee_core error code.

Protected Attributes

- [std::string](#) [message](#)
Message of the exception.

5.31.1 Detailed Description

License error class.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.31.2 Constructor & Destructor Documentation

5.31.2.1 [LicenseError::LicenseError](#) (const [std::string](#) & *mess*)

Creates a new instance.

Parameters:

mess [in] Message of the exception.

5.31.3 Member Function Documentation

5.31.3.1 static [Error](#) [Error::NewError](#) ([int32_t](#) *coreErrorCode*) [*static, inherited*]

Creates a new [Error](#) instance from a sparksee_core error code.

Parameters:

coreErrorCode [in] Sparkseecore error code.

Returns:

Depending on the given sparksee_core error, this may return an [Error](#) instance or an specific [Error](#) subclass instance.

5.31.3.2 const [std::string](#)& [Exception::Message](#) () const [*inherited*]

Gets the message of the exception.

It should be called GetMessage but this is not possible because of a macro defined in Windows.

Returns:

The message of the exception.

5.31.3.3 void Exception::SetMessage (const std::string & mess) [inherited]

Sets the message of the exception.

Parameters:

mess [in] Message.

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

- Exception.h

5.32 NodeExport Class Reference

Stores the node exporting values.

Public Member Functions

- [NodeExport](#) ()
Creates a new instance.
- virtual [~NodeExport](#) ()
Destructor.
- void [SetDefaults](#) ()
Sets to default values.
- const std::wstring & [GetLabel](#) () const
Gets the node label.
- void [SetLabel](#) (const std::wstring &l)
Sets the node label.
- [NodeShape](#) [GetShape](#) () const
Gets the node shape.
- void [SetShape](#) ([NodeShape](#) s)
Sets the node shape.
- [ColorRGB](#) [GetColorRGB](#) () const
Gets the node color.
- void [SetColorRGB](#) ([ColorRGB](#) c)
Sets the node color.
- [ColorRGB](#) [GetLabelColorRGB](#) () const
Gets the node label color.
- void [SetLabelColorRGB](#) ([ColorRGB](#) c)
Sets the node label color.

- `int32_t GetHeight () const`
Gets the node height.
- `void SetHeight (int32_t h)`
Sets the node height.
- `int32_t GetWidth () const`
Gets the node width.
- `void SetWidth (int32_t w)`
Gets the node width.
- `bool_t IsFit () const`
Gets whether the node size is fitted to the label or not.
- `void SetFit (bool_t f)`
Sets the node fit property.
- `int32_t GetFontSize () const`
Gets the node label font size.
- `void SetFontSize (int32_t s)`
Sets the node label font size.

5.32.1 Detailed Description

Stores the node exporting values.

When 'fit' is set to TRUE, then 'height' and 'width' will be ignored.

Some properties may be ignored depending on the exportation type.

Default values are:

Label: "" (empty string).

Shape: Box.

Color: 10863606 (0xa5c3f6).

Label color: 0 (0x000000, Black).

Height: 25px.

Width: 25px.

Fit: TRUE.

Font size: 10.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.32.2 Member Function Documentation

5.32.2.1 `const std::wstring& NodeExport::GetLabel () const` [inline]

Gets the node label.

Returns:

The node label.

5.32.2.2 `void NodeExport::SetLabel (const std::wstring & l)` [inline]

Sets the node label.

Parameters:

l [in] The node label.

5.32.2.3 `NodeShape NodeExport::GetShape () const` [inline]

Gets the node shape.

Returns:

The node shape.

5.32.2.4 `void NodeExport::SetShape (NodeShape s)` [inline]

Sets the node shape.

Parameters:

s [in] The node shape.

5.32.2.5 `ColorRGB NodeExport::GetColorRGB () const` [inline]

Gets the node color.

Returns:

The node color.

5.32.2.6 `void NodeExport::SetColorRGB (ColorRGB c)` [inline]

Sets the node color.

Parameters:

c The node color.

5.32.2.7 ColorRGB NodeExport::GetLabelColorRGB () const [inline]

Gets the node label color.

Returns:

The node label color.

5.32.2.8 void NodeExport::SetLabelColorRGB (ColorRGB c) [inline]

Sets the node label color.

Parameters:

c [in] The node label color.

5.32.2.9 int32_t NodeExport::GetHeight () const [inline]

Gets the node height.

Returns:

The node height in pixels.

5.32.2.10 void NodeExport::SetHeight (int32_t h) [inline]

Sets the node height.

Parameters:

h [in] The node height in pixels.

5.32.2.11 int32_t NodeExport::GetWidth () const [inline]

Gets the node width.

Returns:

The node width in pixels.

5.32.2.12 void NodeExport::SetWidth (int32_t w) [inline]

Gets the node width.

Parameters:

w The node width in pixels.

5.32.2.13 bool_t NodeExport::IsFit () const [inline]

Gets whether the node size is fitted to the label or not.

Returns:

If TRUE, then the node size is fitted to the label, otherwise the size is fixed with the values of 'height' and 'width'.

5.32.2.14 void NodeExport::SetFit (bool_t *f*) [inline]

Sets the node fit property.

Parameters:

f [in] If TRUE, then the node size is fitted to the label ('height' and 'width' will be ignored), otherwise the size is fixed with the values of 'height' and 'width'.

5.32.2.15 int32_t NodeExport::GetFontSize () const [inline]

Gets the node label font size.

Returns:

The node label font size.

5.32.2.16 void NodeExport::SetFontSize (int32_t *s*) [inline]

Sets the node label font size.

Parameters:

s [in] The node label font size.

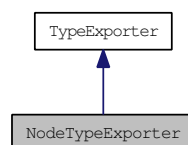
The documentation for this class was generated from the following file:

- Export.h

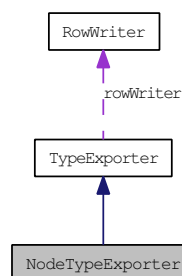
5.33 NodeTypeExporter Class Reference

[NodeTypeExporter](#) class.

Inheritance diagram for NodeTypeExporter:



Collaboration diagram for NodeTypeExporter:



Public Member Functions

- [NodeTypeExporter](#) ()
Creates a new instance.
- [NodeTypeExporter](#) ([RowWriter](#) &rw, [sparksee::gdb::Graph](#) &g, [sparksee::gdb::type_t](#) t, [sparksee::gdb::AttributeList](#) &attrs)
Creates a new instance.
- virtual [~NodeTypeExporter](#) ()
Destructor.
- void [Run](#) () throw ([sparksee::gdb::IOException](#), [sparksee::gdb::Error](#))
See the [TypeExporter](#) class [Run](#) method.
- void [Register](#) ([TypeExporterListener](#) &tel)
Registers a new listener.
- void [SetRowWriter](#) ([RowWriter](#) &rw)
Sets the output data destination.
- void [SetGraph](#) ([sparksee::gdb::Graph](#) &g)
Sets the graph that will be exported.
- void [SetType](#) ([sparksee::gdb::type_t](#) t)
Sets the type to be exported.
- void [SetAttributes](#) ([sparksee::gdb::AttributeList](#) &attrs)
Sets the list of Attributes.
- void [SetFrequency](#) ([sparksee::gdb::int32_t](#) freq)
Sets the frequency of listener notification.
- void [SetHeader](#) ([sparksee::gdb::bool_t](#) header)
Sets the presence of a header row.

Protected Member Functions

- [sparksee::gdb::bool_t CanRun](#) ()
Checks that all the required settings are ready to run.
- void [NotifyListeners](#) (const [TypeExporterEvent](#) &ev)
Notifies progress to all registered listeners.
- void [RunProcess](#) () throw ([sparksee::gdb::IOException](#), [sparksee::gdb::Error](#))
Runs export process.
- void [SetHeadAttribute](#) ([sparksee::gdb::attr_t](#) attr)
Sets the attribute that will be used to get the value to be dumped for the head of the edge.

- void [SetHeadPosition](#) (sparksee::gdb::int32_t pos)
Sets the position (index column) of the head attribute in the exported data.
- void [SetTailAttribute](#) (sparksee::gdb::attr_t attr)
Sets the attribute that will be used to get the value to be dumped for the tail of the edge.
- void [SetTailPosition](#) (sparksee::gdb::int32_t pos)
Sets the position (index column) of the tail attribute in the exported data.

5.33.1 Detailed Description

[NodeTypeExporter](#) class.

Specific [TypeExporter](#) implementation for node types.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.33.2 Constructor & Destructor Documentation

5.33.2.1 [NodeTypeExporter::NodeTypeExporter](#) (RowWriter & *rw*, sparksee::gdb::Graph & *g*, sparksee::gdb::type_t *t*, sparksee::gdb::AttributeList & *attrs*) [inline]

Creates a new instance.

Parameters:

- rw* [in] Output [RowWriter](#).
- g* [in] [Graph](#).
- t* [in] [Type](#) identifier.
- attrs* [in] [Attribute](#) identifiers to be exported.

5.33.3 Member Function Documentation

5.33.3.1 [sparksee::gdb::bool_t TypeExporter::CanRun](#) () [protected, inherited]

Checks that all the required settings are ready to run.

Returns:

Returns true if all the settings are ready.

5.33.3.2 [void TypeExporter::NotifyListeners](#) (const TypeExporterEvent & *ev*) [protected, inherited]

Notifies progress to all registered listeners.

Parameters:

- ev* [in] Progress event to be notified.

5.33.3.3 void TypeExporter::RunProcess () throw (sparksee::gdb::IOException, sparksee::gdb::Error) [protected, inherited]

Runs export process.

Exceptions:

IOException If bad things happen writting to the [RowWriter](#).

5.33.3.4 void TypeExporter::SetHeadAttribute (sparksee::gdb::attr_t attr) [protected, inherited]

Sets the attribute that will be used to get the value to be dumped for the head of the edge.

This method is protected because only the Edge exporters should have it.

Parameters:

attr [in] Head [Attribute](#)

Reimplemented in [EdgeTypeExporter](#).

5.33.3.5 void TypeExporter::SetHeadPosition (sparksee::gdb::int32_t pos) [protected, inherited]

Sets the position (index column) of the head attribute in the exported data.

This method is protected because only the Edge exporters should have it.

Parameters:

pos [in] Head position

Reimplemented in [EdgeTypeExporter](#).

5.33.3.6 void TypeExporter::SetTailAttribute (sparksee::gdb::attr_t attr) [protected, inherited]

Sets the attribute that will be used to get the value to be dumped for the tail of the edge.

This method is protected because only the Edge exporters should have it.

Parameters:

attr [in] Tail [Attribute](#)

Reimplemented in [EdgeTypeExporter](#).

5.33.3.7 void TypeExporter::SetTailPosition (sparksee::gdb::int32_t pos) [protected, inherited]

Sets the position (index column) of the tail attribute in the exported data.

This method is protected because only the Edge exporters should have it.

Parameters:

pos [in] Tail position

Reimplemented in [EdgeTypeExporter](#).

5.33.3.8 void TypeExporter::Register (TypeExporterListener & tel) [inherited]

Registers a new listener.

Parameters:

tel [in] [TypeExporterListener](#) to be registered.

5.33.3.9 void TypeExporter::SetRowWriter (RowWriter & rw) [inherited]

Sets the output data destination.

Parameters:

rw [in] Input [RowWriter](#).

5.33.3.10 void TypeExporter::SetGraph (sparksee::gdb::Graph & g) [inherited]

Sets the graph that will be exported.

Parameters:

g [in] [Graph](#).

5.33.3.11 void TypeExporter::SetType (sparksee::gdb::type_t t) [inherited]

Sets the type to be exported.

Parameters:

t [in] [Type](#) identifier.

5.33.3.12 void TypeExporter::SetAttributes (sparksee::gdb::AttributeList & attrs) [inherited]

Sets the list of Attributes.

Parameters:

attrs [in] [Attribute](#) identifiers to be exported

5.33.3.13 void TypeExporter::SetFrequency (sparksee::gdb::int32_t freq) [inherited]

Sets the frequency of listener notification.

freq [in] Frequency in number of rows managed to notify progress to all listeners

5.33.3.14 void TypeExporter::SetHeader (sparksee::gdb::bool_t header) [inherited]

Sets the presence of a header row.

Parameters:

header [in] If TRUE, a header row is dumped with the name of the attributes.

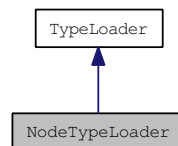
The documentation for this class was generated from the following file:

- NodeTypeExporter.h

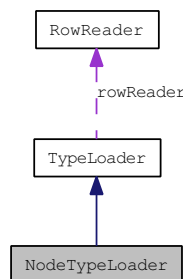
5.34 NodeTypeLoader Class Reference

[NodeTypeLoader](#) class.

Inheritance diagram for NodeTypeLoader:



Collaboration diagram for NodeTypeLoader:



Public Member Functions

- [NodeTypeLoader](#) ()
Creates a new instance.
- [NodeTypeLoader](#) ([RowReader](#) &rr, sparksee::gdb::Graph &g, sparksee::gdb::type_t t, sparksee::gdb::AttributeList &attrs, sparksee::gdb::Int32List &attrsPos)
Creates a new instance.
- virtual [~NodeTypeLoader](#) ()
Destructor.
- void [Run](#) () throw (sparksee::gdb::IOException, sparksee::gdb::Error)
See the [TypeLoader](#) class Run method.

- void [RunTwoPhases](#) () throw (sparksee::gdb::IOException, sparksee::gdb::Error)
See the [TypeLoader](#) class [RunTwoPhases](#) method.
- void [RunNPhases](#) (sparksee::gdb::int32_t partitions) throw (sparksee::gdb::IOException, sparksee::gdb::Error)
See the [TypeLoader](#) class [RunNPhases](#) method.
- void [SetLogError](#) (const std::wstring &path) throw (sparksee::gdb::IOException)
Sets a log error file.
- void [SetLogOff](#) ()
Truns off all the error reporting.
- void [Register](#) (TypeLoaderListener &tel)
Registers a new listener.
- void [SetRowReader](#) (RowReader &rr)
Sets the input data source.
- void [SetGraph](#) (sparksee::gdb::Graph &g)
Sets the graph where the data will be loaded.
- void [SetLocale](#) (const std::wstring &localeStr)
Sets the locale that will be used to read the data.
- void [SetType](#) (sparksee::gdb::type_t t)
Sets the type to be loaded.
- void [SetAttributes](#) (sparksee::gdb::AttributeList &attrs)
Sets the list of Attributes.
- void [SetAttributePositions](#) (sparksee::gdb::Int32List &attrsPos)
Sets the list of attribute positions.
- void [SetTimestampFormat](#) (const std::wstring ×tampFormat)
Sets a specific timestamp format.
- void [SetFrequency](#) (sparksee::gdb::int32_t freq)
Sets the frequency of listener notification.

Protected Types

- enum [Mode](#) {
 [ONE_PHASE](#),
 [TWO_PHASES](#),
 [N_PHASES](#) }
Load can work in different ways.

Protected Member Functions

- `sparksee::gdb::bool_t CanRun ()`
Checks that all the required settings are ready to run.
- `void Run (Mode ph, sparksee::gdb::int32_t par) throw (sparksee::gdb::IOException, sparksee::gdb::Error)`
Runs load process.
- `void NotifyListeners (const TypeLoaderEvent &ev)`
Notifies progress to all registered listeners.
- `void SetHeadAttribute (sparksee::gdb::attr_t attr)`
Sets the attribute that will be used to find the head of the edge.
- `void SetHeadPosition (sparksee::gdb::int32_t pos)`
Sets the position of the head attribute in the source data.
- `void SetTailAttribute (sparksee::gdb::attr_t attr)`
Sets the attribute that will be used to find the tail of the edge.
- `void SetTailPosition (sparksee::gdb::int32_t pos)`
Sets the position of the tail attribute in the source data.

5.34.1 Detailed Description

`NodeTypeLoader` class.

Specific `TypeLoader` implementation for node types.

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.34.2 Member Enumeration Documentation**5.34.2.1 enum TypeLoader::Mode** [protected, inherited]

Load can work in different ways.

Enumerator:

ONE_PHASE Performs the load in a phases.

Load all objects an attributes at the same time.

TWO_PHASES Performs the load in two phases.

Firstly load all objects (and create them if necessary) and secondly loads all the attributes.

Working on this mode it is necessary to build a temporary file.

N_PHASES Performs the load in N phases.

Firstly load all objects (and create them if necessary) and secondly loads all the attributes. But in this case, attributes are loaded one by one. This way, if there are three attributes, then 4 traverses to the [RowReader](#) are necessary.

Working on this mode it is necessary to build a temporary file.

5.34.3 Constructor & Destructor Documentation

5.34.3.1 `NodeTypeLoader::NodeTypeLoader (RowReader & rr, sparksee::gdb::Graph & g, sparksee::gdb::type_t t, sparksee::gdb::AttributeList & attrs, sparksee::gdb::Int32List & attrsPos) [inline]`

Creates a new instance.

Parameters:

rr [in] Input [RowReader](#).
g [in] [Graph](#).
t [in] [Type](#) identifier.
attrs [in] [Attribute](#) identifiers to be loaded.
attrsPos [in] [Attribute](#) positions (column index >=0).

5.34.4 Member Function Documentation

5.34.4.1 `sparksee::gdb::bool_t TypeLoader::CanRun () [protected, inherited]`

Checks that all the required settings are ready to run.

Returns:

Returns true if all the settings are ready.

5.34.4.2 `void TypeLoader::Run (Mode ph, sparksee::gdb::int32_t par) throw (sparksee::gdb::IOException, sparksee::gdb::Error) [protected, inherited]`

Runs load process.

Exceptions:

[IOException](#) If bad things happen reading from the [RowReader](#).

Parameters:

ph [in] The load mode.
par [in] Number of horizontal partitions to perform the load.

5.34.4.3 `void TypeLoader::NotifyListeners (const TypeLoaderEvent & ev) [protected, inherited]`

Notifies progress to all registered listeners.

Parameters:

ev [in] Progress event to be notified.

5.34.4.4 void TypeLoader::SetHeadAttribute (sparksee::gdb::attr_t *attr*) [protected, inherited]

Sets the attribute that will be used to find the head of the edge.

This method is protected because only the Edge loaders should have it.

Parameters:

attr [in] Head [Attribute](#)

Reimplemented in [EdgeTypeLoader](#).

5.34.4.5 void TypeLoader::SetHeadPosition (sparksee::gdb::int32_t *pos*) [protected, inherited]

Sets the position of the head attribute in the source data.

This method is protected because only the Edge loaders should have it.

Parameters:

pos [in] Head position

Reimplemented in [EdgeTypeLoader](#).

5.34.4.6 void TypeLoader::SetTailAttribute (sparksee::gdb::attr_t *attr*) [protected, inherited]

Sets the attribute that will be used to find the tail of the edge.

This method is protected because only the Edge loaders should have it.

Parameters:

attr [in] Tail [Attribute](#)

Reimplemented in [EdgeTypeLoader](#).

5.34.4.7 void TypeLoader::SetTailPosition (sparksee::gdb::int32_t *pos*) [protected, inherited]

Sets the position of the tail attribute in the source data.

This method is protected because only the Edge loaders should have it.

Parameters:

pos [in] Tail position

Reimplemented in [EdgeTypeLoader](#).

5.34.4.8 void TypeLoader::SetLogError (const std::wstring & *path*) throw (sparksee::gdb::IOException) [inherited]

Sets a log error file.

By default errors are thrown as a exception and the load process ends. If a log file is set, errors are logged there and the load process does not stop.

Exceptions:

[IOException](#) If bad things happen opening the file.

5.34.4.9 void TypeLoader::SetLogOff () [inherited]

Turns off all the error reporting.

The log file will not be created and no exceptions for invalid data will be thrown. If you just want to turn off the logs, but abort at the first error what you should do is not call this method and not set a logError file.

5.34.4.10 void TypeLoader::Register (TypeLoaderListener & tel) [inherited]

Registers a new listener.

Parameters:

← *tel* [TypeLoaderListener](#) to be registered.

5.34.4.11 void TypeLoader::SetRowReader (RowReader & rr) [inherited]

Sets the input data source.

Parameters:

rr [in] Input [RowReader](#).

5.34.4.12 void TypeLoader::SetGraph (sparksee::gdb::Graph & g) [inherited]

Sets the graph where the data will be loaded.

Parameters:

g [in] [Graph](#).

5.34.4.13 void TypeLoader::SetLocale (const std::wstring & localeStr) [inherited]

Sets the locale that will be used to read the data.

It should match the locale used in the rowreader.

Parameters:

localeStr [in] The locale string for the read data. See [CSVReader](#).

5.34.4.14 void TypeLoader::SetType (sparksee::gdb::type_t t) [inherited]

Sets the type to be loaded.

Parameters:

t [in] [Type](#) identifier.

5.34.4.15 void TypeLoader::SetAttributes (sparksee::gdb::AttributeList & attrs) [inherited]

Sets the list of Attributes.

Parameters:

attrs [in] [Attribute](#) identifiers to be loaded

5.34.4.16 void TypeLoader::SetAttributePositions (sparksee::gdb::Int32List & attrsPos) [inherited]

Sets the list of attribute positions.

attrsPos [in] [Attribute](#) positions (column index >=0).

5.34.4.17 void TypeLoader::SetFrequency (sparksee::gdb::int32_t freq) [inherited]

Sets the frequency of listener notification.

freq [in] Frequency in number of rows managed to notify progress to all listeners

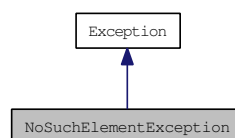
The documentation for this class was generated from the following file:

- NodeTypeLoader.h

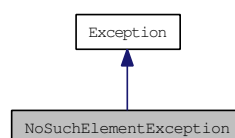
5.35 NoSuchElementException Class Reference

No such element exception class.

Inheritance diagram for NoSuchElementException:



Collaboration diagram for NoSuchElementException:

**Public Member Functions**

- [NoSuchElementException](#) ()
Creates a new instance.
- [NoSuchElementException](#) (const std::string &mess)

Creates a new instance.

- virtual `~NoSuchElementException ()`
Destructor.
- const std::string & `Message ()` const
Gets the message of the exception.
- void `SetMessage (const std::string &mess)`
Sets the message of the exception.

Protected Attributes

- std::string `message`
Message of the exception.

5.35.1 Detailed Description

No such element exception class.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.35.2 Constructor & Destructor Documentation

5.35.2.1 NoSuchElementException::NoSuchElementException (const std::string & mess)

Creates a new instance.

Parameters:

mess [in] Message of the exception.

5.35.3 Member Function Documentation

5.35.3.1 const std::string& Exception::Message () const [inherited]

Gets the message of the exception.

It should be called GetMessage but this is not possible because of a macro defined in Windows.

Returns:

The message of the exception.

5.35.3.2 void Exception::SetMessage (const std::string & mess) [inherited]

Sets the message of the exception.

Parameters:

mess [in] Message.

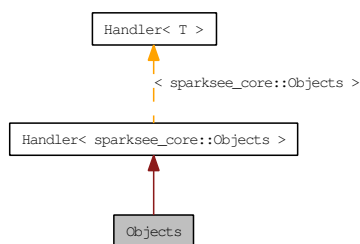
The documentation for this class was generated from the following file:

- Exception.h

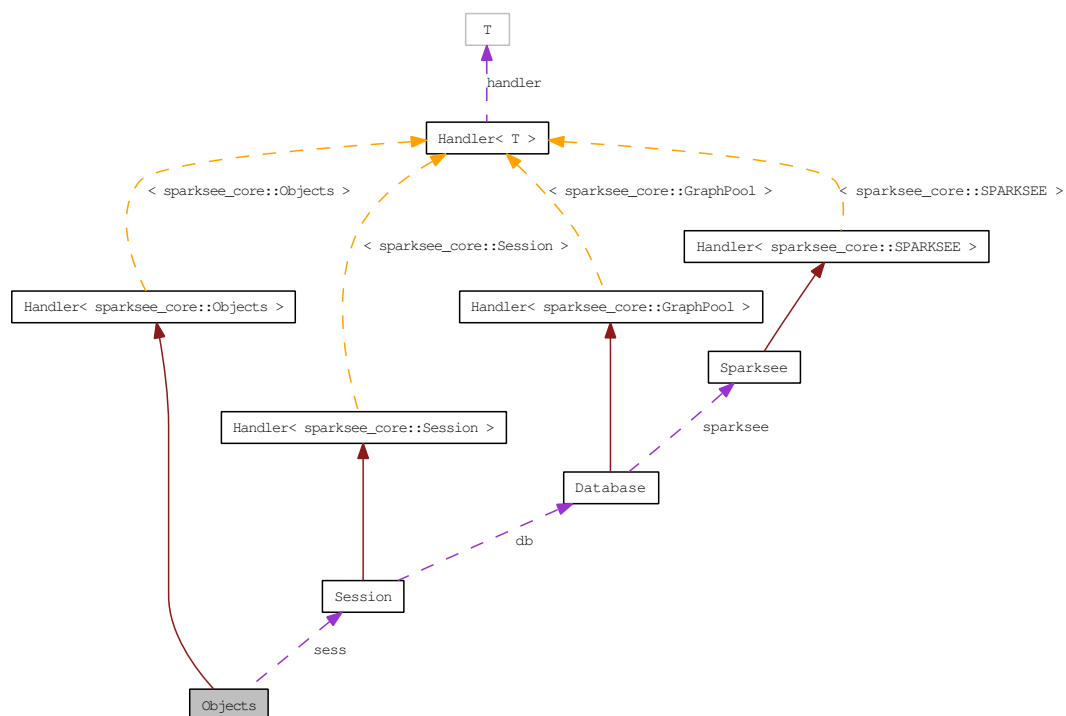
5.36 Objects Class Reference

Object identifier set class.

Inheritance diagram for Objects:



Collaboration diagram for Objects:

**Public Member Functions**

- virtual `~Objects()`

Destructor.

- **Objects * Copy ()**
*Creates a new **Objects** instance as a copy of the given one.*
- **int64_t Count ()**
Gets the number of elements into the collection.
- **bool_t Add (oid_t e)**
Adds an element into the collection.
- **bool_t Exists (oid_t e)**
Gets if the given element exists into the collection.
- **oid_t Any () throw (sparksee::gdb::NoSuchElementException, sparksee::gdb::Error)**
Gets an element from the collection.
- **bool_t Remove (oid_t e)**
Removes an element from the collection.
- **void Clear ()**
Clears the collection removing all its elements.
- **int64_t Union (Objects *objs)**
Performs the union operation.
- **int64_t Intersection (Objects *objs)**
Performs the intersection operation.
- **int64_t Difference (Objects *objs)**
Performs the difference operation.
- **bool_t Equals (Objects *objs)**
*Checks if the given **Objects** contains the same information.*
- **bool_t Contains (Objects *objs)**
Check if this objects contains the other one.
- **int64_t Copy (Objects *objs)**
Performs the copy operation.
- **Objects * Sample (Objects *exclude, int64_t samples)**
*Creates a new **Objects** instance which is a sample of the calling one.*
- **ObjectsIterator * Iterator ()**
*Gets an **ObjectsIterator**.*
- **ObjectsIterator * IteratorFromIndex (int64_t index)**
*Gets an **ObjectsIterator** skipping index elements.*

- `ObjectsIterator * IteratorFromElement (oid_t e)`
Gets an `ObjectsIterator` starting from the given element.

Static Public Member Functions

- static `Objects * CombineUnion (Objects *objs1, Objects *objs2)`
Creates a new `Objects` instance which is the union of the two given.
- static `Objects * CombineIntersection (Objects *objs1, Objects *objs2)`
Creates a new `Objects` instance which is the intersection of the two given.
- static `Objects * CombineDifference (Objects *objs1, Objects *objs2)`
Creates a new `Objects` instance which is the difference of the two given.

Static Public Attributes

- static const `oid_t InvalidOID`
Invalid OID constant.

Friends

- class `Session`
- class `Graph`
- class `ObjectsIterator`

5.36.1 Detailed Description

Object identifier set class.

It stores a collection of `Sparksee` object identifiers as a set. As a set, there is no order and no duplicated elements.

This class should be used just to store large collections. Otherwise, it is strongly recommended to use common classes from the language API.

This class is not thread-safe.

`ObjectsIterator` must be used to traverse all the elements into the set.

When the `Objects` instance is closed, it closes all existing and non-closed `ObjectsIterator` instances too.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.36.2 Member Function Documentation

5.36.2.1 `Objects* Objects::Copy ()`

Creates a new `Objects` instance as a copy of the given one.

Returns:

The new [Objects](#) instance.

5.36.2.2 int64_t Objects::Count ()

Gets the number of elements into the collection.

Returns:

The number of elements into the collection.

5.36.2.3 bool_t Objects::Add (oid_t e)

Adds an element into the collection.

Parameters:

e [in] Element to be added.

Returns:

TRUE if the element is added, FALSE if the element was already into the collection.

5.36.2.4 bool_t Objects::Exists (oid_t e)

Gets if the given element exists into the collection.

Parameters:

e [in] Element.

Returns:

TRUE if the element exists into the collection, FALSE otherwise.

5.36.2.5 oid_t Objects::Any () throw (sparksee::gdb::NoSuchElementException, sparksee::gdb::Error)

Gets an element from the collection.

Returns:

Any element from the collection.

Exceptions:

[*NoSuchElementException*](#) whether the collection is empty.

5.36.2.6 bool_t Objects::Remove (oid_t *e*)

Removes an element from the collection.

Parameters:

e [in] Element to be removed.

Returns:

TRUE if the element is removed, FALSE if the element was not into the collection.

5.36.2.7 int64_t Objects::Union (Objects * *objs*)

Performs the union operation.

This adds all existing elements of the given [Objects](#) instance to the [Objects](#) calling instance

Parameters:

objs [in] [Objects](#) instance.

Returns:

Number of elements into the collection once the operation has been executed.

5.36.2.8 int64_t Objects::Intersection (Objects * *objs*)

Performs the intersection operation.

Updates the [Objects](#) calling instance setting those existing elements at both two collections and removing all others.

Parameters:

objs [in] [Objects](#) instance.

Returns:

Number of elements into the collection once the operation has been executed.

5.36.2.9 int64_t Objects::Difference (Objects * *objs*)

Performs the difference operation.

This updates the [Objects](#) calling instance removing those existing elements at the given [Objects](#) instance.

Parameters:

objs [in] [Objects](#) instance.

Returns:

Number of elements into the collection once the operation has been executed.

5.36.2.10 bool_t Objects::Equals (Objects * *objs*)

Checks if the given [Objects](#) contains the same information.

Parameters:

objs [in] [Objects](#) instance.

Returns:

True if the objects are equal or false otherwise.

5.36.2.11 bool_t Objects::Contains (Objects * *objs*)

Check if this objects contains the other one.

Parameters:

objs [Objects](#) collection.

Returns:

True if it contains the given object.

5.36.2.12 static Objects* Objects::CombineUnion (Objects * *objs1*, Objects * *objs2*) [static]

Creates a new [Objects](#) instance which is the union of the two given.

Two given [Objects](#) belong to the same [Session](#).

Parameters:

objs1 [in] [Objects](#) instance.

objs2 [in] [Objects](#) instance.

Returns:

New [Objects](#) instance.

5.36.2.13 static Objects* Objects::CombineIntersection (Objects * *objs1*, Objects * *objs2*) [static]

Creates a new [Objects](#) instance which is the intersection of the two given.

Two given [Objects](#) belong to the same [Session](#).

Parameters:

objs1 [in] [Objects](#) instance.

objs2 [in] [Objects](#) instance.

Returns:

New [Objects](#) instance.

5.36.2.14 static Objects* Objects::CombineDifference (Objects * *objs1*, Objects * *objs2*)
[static]

Creates a new [Objects](#) instance which is the difference of the two given.

Two given [Objects](#) belong to the same [Session](#).

Parameters:

objs1 [in] [Objects](#) instance.

objs2 [in] [Objects](#) instance.

Returns:

New [Objects](#) instance.

5.36.2.15 int64_t Objects::Copy (Objects * *objs*)

Performs the copy operation.

This updates the [Objects](#) calling instance and copies the given [Objects](#) instance.

Parameters:

objs [in] [Objects](#) instance.

Returns:

Number of elements into the collection once the operation has been executed.

5.36.2.16 Objects* Objects::Sample (Objects * *exclude*, int64_t *samples*)

Creates a new [Objects](#) instance which is a sample of the calling one.

Parameters:

exclude [in] If not NULL, elements into this collection will be excluded from the resulting one.

samples [in] Number of elements into the resulting collection.

Returns:

Sample collection.

5.36.2.17 ObjectsIterator* Objects::Iterator ()

Gets an [ObjectsIterator](#).

Returns:

[ObjectsIterator](#) instance.

5.36.2.18 ObjectsIterator* Objects::IteratorFromIndex (int64_t *index*)

Gets an [ObjectsIterator](#) skipping index elements.

[Objects](#) collection has no order, so this method is implementation-dependent.

Parameters:

index [in] The number of elements to skip from the beginning. It must be in the range [0..Size).

Returns:

[ObjectsIterator](#) instance.

5.36.2.19 ObjectsIterator* Objects::IteratorFromElement (oid_t *e*)

Gets an [ObjectsIterator](#) starting from the given element.

[Objects](#) collection has no order, so this method is implementation-dependent.

Parameters:

e [in] The first element to traverse in the resulting [ObjectsIterator](#) instance.

Returns:

[ObjectsIterator](#) instance.

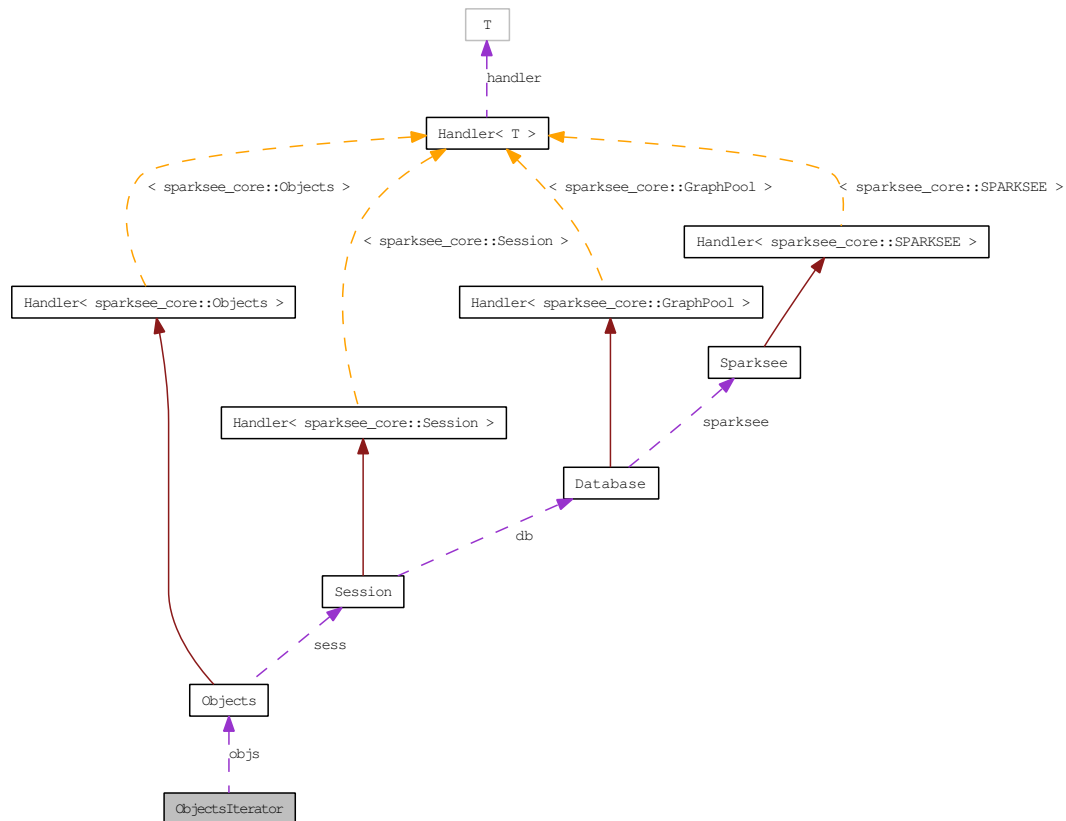
The documentation for this class was generated from the following file:

- [Objects.h](#)

5.37 ObjectsIterator Class Reference

[ObjectsIterator](#) class.

Collaboration diagram for ObjectsIterator:



Public Member Functions

- virtual [~ObjectsIterator](#) ()
Destructor.
- [bool_t HasNext](#) ()
Gets if there are more elements to traverse.
- [oid_t Next](#) ()
Gets the next element to traverse.

Friends

- class [Objects](#)

5.37.1 Detailed Description

[ObjectsIterator](#) class.

Iterator to traverse all the object identifiers from an [Objects](#) instance.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.37.2 Member Function Documentation**5.37.2.1 bool_t ObjectsIterator::HasNext ()**

Gets if there are more elements to traverse.

Returns:

TRUE if there are more elements to traverse, FALSE otherwise.

5.37.2.2 oid_t ObjectsIterator::Next ()

Gets the next element to traverse.

Returns:

The next element.

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

- ObjectsIterator.h

5.38 OIDList Class Reference

[Sparksee](#) object identifier list.

Public Member Functions

- [int32_t Count](#) () const
Number of elements in the list.
- [OIDListIterator * Iterator](#) ()
Gets a new [OIDListIterator](#).
- [OIDList](#) ()
Constructor.
- [OIDList](#) ([int32_t](#) N)
Constructor.
- [OIDList](#) (const std::vector< [oid_t](#) > &v)
Constructor.
- [~OIDList](#) ()
Destructor.
- void [Add](#) ([oid_t](#) attr)

Adds a *Sparksee* object identifier at the end of the list.

- void **Set** (int32_t pos, oid_t oid)

Sets a *Sparksee* object identifier at the specified position of the list.

- void **Clear** ()

Clears the list.

5.38.1 Detailed Description

Sparksee object identifier list.

It stores a *Sparksee* object identifier list.

Use *OIDListIterator* to access all elements into this collection.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.38.2 Constructor & Destructor Documentation

5.38.2.1 OIDList::OIDList ()

Constructor.

This creates an empty list.

5.38.2.2 OIDList::OIDList (int32_t N)

Constructor.

This creates a list with N invalid oids.

5.38.2.3 OIDList::OIDList (const std::vector< oid_t > & v)

Constructor.

Parameters:

v [in] Vector.

5.38.3 Member Function Documentation

5.38.3.1 int32_t OIDList::Count () const [inline]

Number of elements in the list.

Returns:

Number of elements in the list.

5.38.3.2 `OIDListIterator*` `OIDList::Iterator ()`

Gets a new [OIDListIterator](#).

Returns:

[OIDListIterator](#) instance.

5.38.3.3 `void` `OIDList::Add (oid_t attr)` `[inline]`

Adds a [Sparksee](#) object identifier at the end of the list.

Parameters:

attr [in] [Sparksee](#) object identifier.

5.38.3.4 `void` `OIDList::Set (int32_t pos, oid_t oid)` `[inline]`

Sets a [Sparksee](#) object identifier at the specified position of the list.

Parameters:

pos [in] List position [0..[Count\(\)](#)-1].

oid [in] [Sparksee](#) object identifier.

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

- [Graph_data.h](#)

5.39 `OIDListIterator` Class Reference

[OIDList](#) iterator class.

Public Member Functions

- [~OIDListIterator \(\)](#)
Destructor.
- [oid_t Next \(\)](#)
Moves to the next element.
- [bool_t HasNext \(\)](#)
Gets if there are more elements.

Friends

- class [OIDList](#)

5.39.1 Detailed Description

[OIDList](#) iterator class.

Iterator to traverse all the [Sparksee](#) object identifier into a [OIDList](#) instance.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.39.2 Member Function Documentation

5.39.2.1 `oid_t OIDListIterator::Next ()` [inline]

Moves to the next element.

Returns:

The next element.

5.39.2.2 `bool_t OIDListIterator::HasNext ()` [inline]

Gets if there are more elements.

Returns:

TRUE if there are more elements, FALSE otherwise.

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

- `Graph_data.h`

5.40 Platform Class Reference

[Platform](#) class.

Static Public Member Functions

- static void [GetStatistics](#) ([PlatformStatistics](#) &stats)
Gets platform data and statistics.

5.40.1 Detailed Description

[Platform](#) class.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.40.2 Member Function Documentation

5.40.2.1 static void Platform::GetStatistics (PlatformStatistics & stats) [static]

Gets platform data and statistics.

Parameters:

stats [in|out] This updates the given [PlatformStatistics](#).

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

- common.h

5.41 PlatformStatistics Class Reference

[Platform](#) data and statistics.

Public Member Functions

- [PlatformStatistics](#) ()
Creates a new instance setting all values to 0.
- [int32_t](#) GetNumCPUs () const
Gets the number of CPUs.
- [int64_t](#) GetRealTime () const
Gets time in microseconds (since epoch).
- [int64_t](#) GetUserTime () const
Gets CPU user time.
- [int64_t](#) GetSystemTime () const
Gets CPU system time.
- [int64_t](#) GetTotalMem () const
Gets physical memory size in Bytes.
- [int64_t](#) GetAvailableMem () const
Gets available (free) memory size in Bytes.

Friends

- class [Platform](#)

5.41.1 Detailed Description

[Platform](#) data and statistics.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.41.2 Member Function Documentation

5.41.2.1 `int32_t PlatformStatistics::GetNumCPUs () const` `[inline]`

Gets the number of CPUs.

Returns:

The number of CPUs.

5.41.2.2 `int64_t PlatformStatistics::GetRealTime () const` `[inline]`

Gets time in microseconds (since epoch).

Returns:

Time in microseconds (since epoch).

5.41.2.3 `int64_t PlatformStatistics::GetUserTime () const` `[inline]`

Gets CPU user time.

Returns:

CPU user time.

5.41.2.4 `int64_t PlatformStatistics::GetSystemTime () const` `[inline]`

Gets CPU system time.

Returns:

CPU system time.

5.41.2.5 `int64_t PlatformStatistics::GetTotalMem () const` `[inline]`

Gets physical memory size in Bytes.

Returns:

Physical memory size in Bytes.

5.41.2.6 `int64_t PlatformStatistics::GetAvailableMem () const` `[inline]`

Gets avialable (free) memory size in Bytes.

Returns:

Avialable (free) memory size in Bytes.

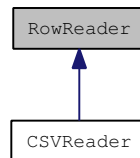
The documentation for this class was generated from the following file:

- common.h

5.42 RowReader Class Reference

[RowReader](#) interface.

Inheritance diagram for RowReader:



Public Member Functions

- virtual [sparksee::gdb::bool_t Reset](#) ()=0 throw (sparksee::gdb::IOException)
Moves the reader to the beginning.
- virtual [sparksee::gdb::bool_t Read](#) (sparksee::gdb::StringList &row)=0 throw (sparksee::gdb::IOException)
Reads the next row as a string array.
- virtual [sparksee::gdb::int32_t GetRow](#) ()=0 throw (sparksee::gdb::IOException)
The row number for the current row.
- virtual void [Close](#) ()=0 throw (sparksee::gdb::IOException)
Closes the reader.
- virtual [~RowReader](#) ()
Destructor.

Protected Member Functions

- [RowReader](#) ()
Empty constructor.

5.42.1 Detailed Description

[RowReader](#) interface.

Common interface for those readers which get the data as an string array.

It works as follows: perform as many read operations as necessary and call close once at the end. Once close is called no more read operations can be executed.

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.42.2 Constructor & Destructor Documentation

5.42.2.1 RowReader::RowReader () [inline, protected]

Empty constructor.

Protected because no one should instantiate a [RowReader](#). Just inherited classes can use this empty constructor.

5.42.3 Member Function Documentation

5.42.3.1 virtual sparksee::gdb::bool_t RowReader::Reset () throw (sparksee::gdb::IOException) [pure virtual]

Moves the reader to the beginning.

Restarts the reader.

Returns:

`true` if the reader can be restarted, `false` otherwise.

Exceptions:

[IOException](#) If bad things happen during the restart.

Implemented in [CSVReader](#).

5.42.3.2 virtual sparksee::gdb::bool_t RowReader::Read (sparksee::gdb::StringList & row) throw (sparksee::gdb::IOException) [pure virtual]

Reads the next row as a string array.

Parameters:

row [out] A string list with each comma-separated element as a separate entry.

Returns:

Returns true if a row had been read or false otherwise.

Exceptions:

[IOException](#) If bad things happen during the read.

Implemented in [CSVReader](#).

5.42.3.3 virtual sparksee::gdb::int32_t RowReader::GetRow () throw (sparksee::gdb::IOException) [pure virtual]

The row number for the current row.

Returns:

The current row number; 0 if there is no current row.

Exceptions:

[IOException](#) If it fails.

Implemented in [CSVReader](#).

5.42.3.4 `virtual void RowReader::Close () throw (sparksee::gdb::IOException) [pure virtual]`

Closes the reader.

Exceptions:

[IOException](#) If the close fails.

Implemented in [CSVReader](#).

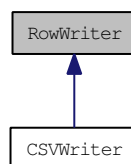
The documentation for this class was generated from the following file:

- RowReader.h

5.43 RowWriter Class Reference

[RowWriter](#) interface.

Inheritance diagram for RowWriter:



Public Member Functions

- virtual void [Write](#) (sparksee::gdb::StringList &row)=0 throw (sparksee::gdb::IOException, sparksee::gdb::Error)
Writes the next row.
- virtual void [Close](#) ()=0 throw (sparksee::gdb::IOException, sparksee::gdb::Error)
Closes the writer.
- virtual [~RowWriter](#) ()
Destructor.

Protected Member Functions

- [RowWriter](#) ()
Empty constructor.

5.43.1 Detailed Description

[RowWriter](#) interface.

Common interface for those writers which dump the data from an string array.

It works as follows: perform as many write operations as necessary and call close once at the end. Once close is called no more write operations can be executed.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.43.2 Constructor & Destructor Documentation**5.43.2.1 RowWriter::RowWriter ()** [inline, protected]

Empty constructor.

Protected because no one should instantiate a [RowWriter](#). Just inherited classes can use this empty constructor.

5.43.3 Member Function Documentation**5.43.3.1 virtual void RowWriter::Write (sparksee::gdb::StringList & row) throw (sparksee::gdb::IOException, sparksee::gdb::Error)** [pure virtual]

Writes the next row.

Parameters:

row [in] Row of data.

Exceptions:

[IOException](#) If bad things happen during the write.

Implemented in [CSVWriter](#).

5.43.3.2 virtual void RowWriter::Close () throw (sparksee::gdb::IOException, sparksee::gdb::Error) [pure virtual]

Closes the writer.

Exceptions:

[IOException](#) If the close fails.

Implemented in [CSVWriter](#).

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

- RowWriter.h

5.44 ScriptParser Class Reference

[ScriptParser](#).

Public Member Functions

- [ScriptParser \(\)](#)
Constructor.
- virtual [~ScriptParser \(\)](#)
Destructor.
- void [SetOutputLog](#) (const std::wstring &path) throw (sparksee::gdb::IOException)
Sets the output log.
- void [SetErrorLog](#) (const std::wstring &path) throw (sparksee::gdb::IOException)
Sets the error log.
- [sparksee::gdb::bool_t Parse](#) (sparksee_core::FileReader *fileReader, [sparksee::gdb::bool_t](#) execute)
Parser the given input stream.
- [sparksee::gdb::bool_t Parse](#) (const std::wstring &path, [sparksee::gdb::bool_t](#) execute, const std::wstring &localeStr) throw (sparksee::gdb::IOException)
Parses the given input file.

Static Public Member Functions

- static void [GenerateSchemaScript](#) (const std::wstring &path, sparksee::gdb::Database &db) throw (sparksee::gdb::IOException)
Writes an script with the schema definition for the given database.

5.44.1 Detailed Description

[ScriptParser](#).

The [ScriptParser](#) can create schemas and load data from a set of commands in a sparksee script.

A SPARKSEE script contains an ordered list of commands. [ScriptParser](#) will execute each one of them in order. Commands may create schemas, define nodes and edges, and load data into a previous defined SPARKSEE schema.

Check out the 'Scripting' chapter in the SPARKSEE User Manual for a comprehensive explanation on the grammar of the SPARKSEE commands and how they work.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.44.2 Member Function Documentation

5.44.2.1 void ScriptParser::SetOutputLog (const std::wstring & path) throw (sparksee::gdb::IOException)

Sets the output log.

If not set, output log corresponds to standard output.

Parameters:

path [in] Path of the output log.

Exceptions:

[*IOException*](#) If bad things happen opening the file.

5.44.2.2 void ScriptParser::SetErrorLog (const std::wstring & *path*) throw (sparksee::gdb::IOException)

Sets the error log.

If not set, error log corresponds to standard error output.

Parameters:

path [in] Path of the error log.

Exceptions:

[*IOException*](#) If bad things happen opening the file.

5.44.2.3 sparksee::gdb::bool_t ScriptParser::Parse (sparksee_core::FileReader * *fileReader*, sparksee::gdb::bool_t *execute*)

Parser the given input stream.

Parameters:

fileReader [in] Input file reader.

execute [in] If TRUE the script is executed, if FALSE it is just parsed.

Returns:

TRUE if ok, FALSE in case of error.

5.44.2.4 sparksee::gdb::bool_t ScriptParser::Parse (const std::wstring & *path*, sparksee::gdb::bool_t *execute*, const std::wstring & *localeStr*) throw (sparksee::gdb::IOException)

Parses the given input file.

Parameters:

path [in] Input file path.

execute [in] If TRUE the script is executed, if FALSE it is just parsed.

localeStr [in] The locale string for reading the input file. See [CSVReader](#).

Returns:

TRUE if ok, FALSE in case of error.

Exceptions:

[*IOException*](#) If bad things happen opening the file.

5.44.2.5 `static void ScriptParser::GenerateSchemaScript (const std::wstring & path, sparksee::gdb::Database & db) throw (sparksee::gdb::IOException) [static]`

Writes an script with the schema definition for the given database.

Parameters:

path [in] Filename of the script to be written.

db [in] [Database](#).

Exceptions:

[IOException](#) If bad things happen opening or writing the file.

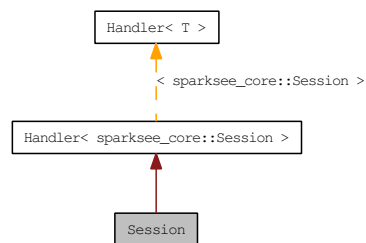
The documentation for this class was generated from the following file:

- ScriptParser.h

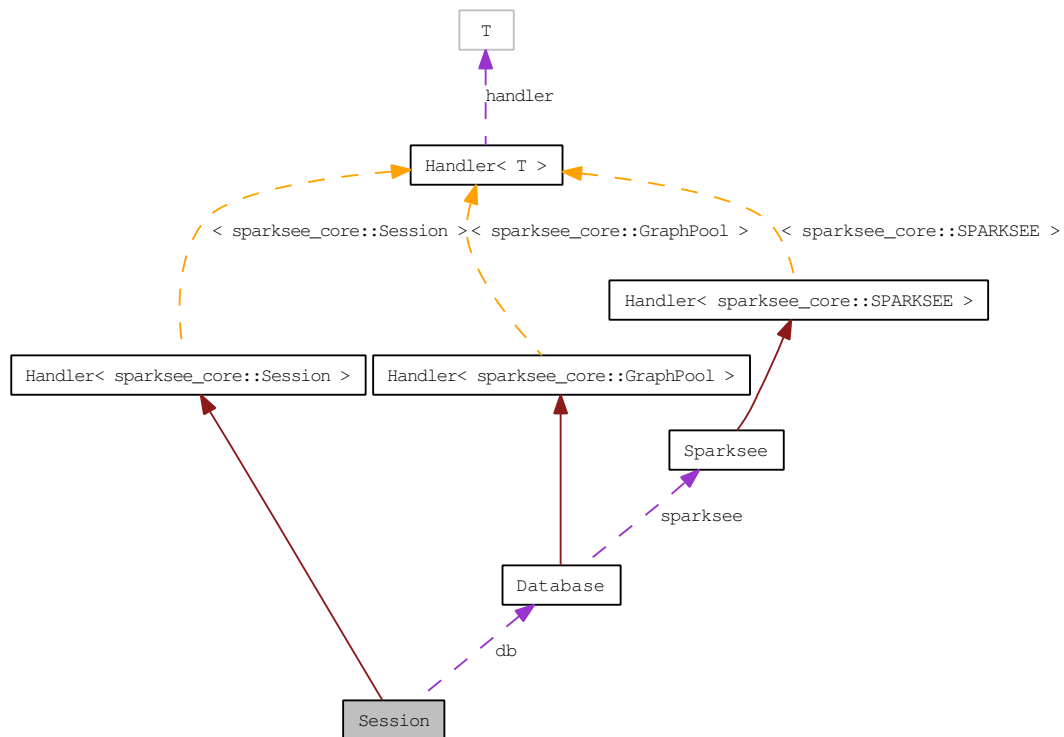
5.45 Session Class Reference

[Session](#) class.

Inheritance diagram for Session:



Collaboration diagram for Session:



Public Member Functions

- virtual `~Session ()`
Destructor.
- `Graph * GetGraph ()`
Gets the `Graph` instance.
- `Objects * NewObjects ()`
Creates a new `Objects` instance.
- void `Begin ()`
Begins a transaction.
- void `Commit ()`
Commits a transaction.

Friends

- class `Database`
- class `Graph`
- class `Objects`
- class `ObjectsIterator`

- class [Values](#)
- class [ValuesIterator](#)
- class [TextStream](#)

5.45.1 Detailed Description

[Session](#) class.

A [Session](#) is a stateful period of activity of a user with the [Database](#).

All the manipulation of a [Database](#) must be enclosed into a [Session](#). A [Session](#) can be initiated from a [Database](#) instance and allows for getting a [Graph](#) instance which represents the persistent graph (the graph database).

Also, temporary data is associated to the [Session](#), thus when a [Session](#) is closed, all the temporary data associated to the [Session](#) is removed too. [Objects](#) or [Values](#) instances or even session attributes are an example of temporary data.

Moreover, a [Session](#) is exclusive for a thread, thus if it is shared among threads results may be fatal or unexpected.

Check out the 'Processing' and 'Transactions' sections in the SPARKSEE User Manual for details about how Sessions work and the use of transactions.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.45.2 Member Function Documentation

5.45.2.1 [Graph*](#) [Session::GetGraph \(\)](#)

Gets the [Graph](#) instance.

Returns:

The [Graph](#) instance.

5.45.2.2 [Objects*](#) [Session::NewObjects \(\)](#)

Creates a new [Objects](#) instance.

Returns:

The new [Objects](#) instance.

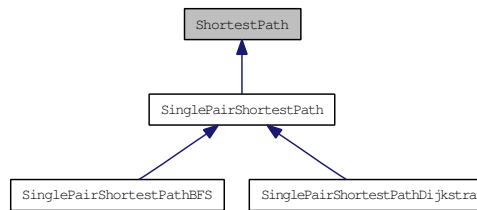
The documentation for this class was generated from the following file:

- [Session.h](#)

5.46 ShortestPath Class Reference

[ShortestPath](#) class.

Inheritance diagram for ShortestPath:



Public Member Functions

- void [SetMaximumHops](#) (sparksee::gdb::int32_t maxhops)
Sets the maximum hops restriction.
- virtual void [AddEdgeType](#) (sparksee::gdb::type_t t, sparksee::gdb::EdgesDirection d)
Allows for traversing edges of the given type.
- virtual void [AddAllEdgeTypes](#) (sparksee::gdb::EdgesDirection d)
Allows for traversing all edge types of the graph.
- virtual void [AddNodeType](#) (sparksee::gdb::type_t t)
Allows for traversing nodes of the given type.
- virtual void [AddAllNodeTypes](#) ()
Allows for traversing all node types of the graph.
- virtual void [ExcludeNodes](#) (sparksee::gdb::Objects &nodes)
Set which nodes can't be used.
- virtual void [ExcludeEdges](#) (sparksee::gdb::Objects &edges)
Set which edges can't be used.
- virtual void [Run](#) ()=0
Runs the algorithm.
- virtual [~ShortestPath](#) ()
Destructor.

Protected Member Functions

- [ShortestPath](#) (sparksee::gdb::Session &s)
Creates a new instance.
- void [AssertEdgeType](#) (sparksee::gdb::type_t edgetype)
Check that the given edge type is valid.
- void [AssertAddedEdges](#) ()

Check that edges had been added.

- void [AssertAddedNodes](#) ()
Check that nodes had been added.
- void [AssertNodeType](#) (sparksee::gdb::type_t nodetype)
Check that the given node type is valid.
- sparksee::gdb::bool_t [IsNodeTypeAllowed](#) (sparksee::gdb::oid_t nodeId)
Check if the given node has an allowed type.
- void [AssertNotComputed](#) ()
Check that the shortest path had not been calculated yet.
- void [AssertComputed](#) ()
Check that the shortest path had been calculated.
- sparksee::gdb::bool_t [IsNodeExcluded](#) (sparksee::gdb::oid_t node)
Check if the given node is forbidden.
- sparksee::gdb::bool_t [IsEdgeExcluded](#) (sparksee::gdb::oid_t edge)
Check if the given edge is forbidden.

Protected Attributes

- sparksee::gdb::Session * [sess](#)
Session.
- sparksee::gdb::Graph * [graph](#)
Graph.
- std::map< [sparksee::gdb::type_t](#), [sparksee::gdb::EdgesDirection](#) > [edgeTypes](#)
Allowed edge types.
- std::vector< [sparksee::gdb::type_t](#) > [nodeTypes](#)
Allowed node types.
- [sparksee::gdb::int32_t](#) [maxHops](#)
Maximum hops restriction.
- [sparksee::gdb::bool_t](#) [computed](#)
True if the shortest path has been calculated.
- sparksee::gdb::Objects * [excludedNodes](#)
The set of excluded nodes.
- sparksee::gdb::Objects * [excludedEdges](#)
The set of excluded edges.

- `sparksee::gdb::bool_t areAllNodeTypesAllowed`

True if all the node types are allowed.

5.46.1 Detailed Description

`ShortestPath` class.

Classes implementing this abstract class solve the shortest path problem in a graph.

The user must set which node and edge types can be used for the traversal.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.46.2 Constructor & Destructor Documentation

5.46.2.1 `ShortestPath::ShortestPath (sparksee::gdb::Session & s) [protected]`

Creates a new instance.

Parameters:

s [in] `Session` to get the graph from and perform traversal.

5.46.3 Member Function Documentation

5.46.3.1 `void ShortestPath::SetMaximumHops (sparksee::gdb::int32_t maxhops)`

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

Parameters:

m [in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

5.46.3.2 `virtual void ShortestPath::AddEdgeType (sparksee::gdb::type_t t, sparksee::gdb::EdgesDirection d) [virtual]`

Allows for traversing edges of the given type.

Parameters:

t [in] Edge type.

d [in] Edge direction.

5.46.3.3 `virtual void ShortestPath::AddAllEdgeTypes (sparksee::gdb::EdgesDirection d)` [virtual]

Allows for traversing all edge types of the graph.

Parameters:

d [in] Edge direction.

5.46.3.4 `virtual void ShortestPath::ExcludeNodes (sparksee::gdb::Objects & nodes)` [virtual]

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters:

nodes [in] A set of node identifiers that must be kept intact until the destruction of the class.

5.46.3.5 `virtual void ShortestPath::ExcludeEdges (sparksee::gdb::Objects & edges)` [virtual]

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters:

edges [in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.46.3.6 `virtual void ShortestPath::Run ()` [pure virtual]

Runs the algorithm.

This method can only be called once.

Implemented in [SinglePairShortestPathBFS](#), and [SinglePairShortestPathDijkstra](#).

5.46.4 Member Data Documentation

5.46.4.1 `sparksee::gdb::int32_t ShortestPath::maxHops` [protected]

Maximum hops restriction.

It must be positive or zero. Zero means unlimited.

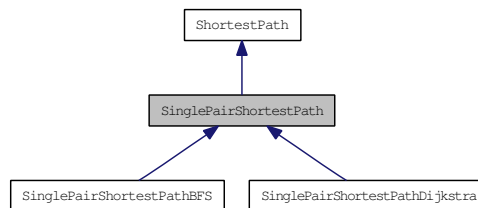
The documentation for this class was generated from the following file:

- ShortestPath.h

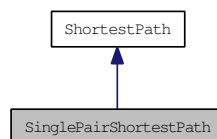
5.47 SinglePairShortestPath Class Reference

[SinglePairShortestPath](#) class.

Inheritance diagram for SinglePairShortestPath:



Collaboration diagram for SinglePairShortestPath:



Public Member Functions

- virtual `sparksee::gdb::OIDList * GetPathAsNodes ()=0`
Gets the shortest path between the source node and the destination node as an ordered set of nodes.
- virtual `sparksee::gdb::OIDList * GetPathAsEdges ()=0`
Gets the shortest path between the source node and the destination node as an ordered set of edges.
- virtual `sparksee::gdb::double64_t GetCost ()=0`
Gets the cost of the shortest path.
- virtual `sparksee::gdb::bool_t Exists ()`
Returns TRUE If a path exists or FALSE otherwise.
- virtual `~SinglePairShortestPath ()`
Destructor.
- void `SetMaximumHops (sparksee::gdb::int32_t maxhops)`
Sets the maximum hops restriction.
- virtual void `AddEdgeType (sparksee::gdb::type_t t, sparksee::gdb::EdgesDirection d)`
Allows for traversing edges of the given type.
- virtual void `AddAllEdgeTypes (sparksee::gdb::EdgesDirection d)`
Allows for traversing all edge types of the graph.
- virtual void `AddNodeType (sparksee::gdb::type_t t)`
Allows for traversing nodes of the given type.

- virtual void [AddAllNodeTypes](#) ()
Allows for traversing all node types of the graph.
- virtual void [ExcludeNodes](#) (sparksee::gdb::Objects &nodes)
Set which nodes can't be used.
- virtual void [ExcludeEdges](#) (sparksee::gdb::Objects &edges)
Set which edges can't be used.
- virtual void [Run](#) ()=0
Runs the algorithm.

Protected Member Functions

- [SinglePairShortestPath](#) (sparksee::gdb::Session &s, [sparksee::gdb::oid_t](#) src, [sparksee::gdb::oid_t](#) dst)
Creates a new instance.
- void [AssertEdgeType](#) (sparksee::gdb::type_t edgetype)
Check that the given edge type is valid.
- void [AssertAddedEdges](#) ()
Check that edges had been added.
- void [AssertAddedNodes](#) ()
Check that nodes had been added.
- void [AssertNodeType](#) (sparksee::gdb::type_t nodetype)
Check that the given node type is valid.
- [sparksee::gdb::bool_t IsNodeTypeAllowed](#) (sparksee::gdb::oid_t nodeId)
Check if the given node has an allowed type.
- void [AssertNotComputed](#) ()
Check that the shortest path had not been calculated yet.
- void [AssertComputed](#) ()
Check that the shortest path had been calculated.
- [sparksee::gdb::bool_t IsNodeExcluded](#) (sparksee::gdb::oid_t node)
Check if the given node is forbidden.
- [sparksee::gdb::bool_t IsEdgeExcluded](#) (sparksee::gdb::oid_t edge)
Check if the given edge is forbidden.

Protected Attributes

- [sparksee::gdb::oid_t source](#)
Source node.
- [sparksee::gdb::oid_t destination](#)
Destination node.
- [sparksee::gdb::OIDList * pathAsNodes](#)
Ordered set of node identifiers representing the shortest path.
- [sparksee::gdb::OIDList * pathAsEdges](#)
Ordered set of edge identifiers representing the shortest path.
- [sparksee::gdb::Session * sess](#)
Session.
- [sparksee::gdb::Graph * graph](#)
Graph.
- [std::map< sparksee::gdb::type_t, sparksee::gdb::EdgesDirection > edgeTypes](#)
Allowed edge types.
- [std::vector< sparksee::gdb::type_t > nodeTypes](#)
Allowed node types.
- [sparksee::gdb::int32_t maxHops](#)
Maximum hops restriction.
- [sparksee::gdb::bool_t computed](#)
True if the shortest path has been calculated.
- [sparksee::gdb::Objects * excludedNodes](#)
The set of excluded nodes.
- [sparksee::gdb::Objects * excludedEdges](#)
The set of excluded edges.
- [sparksee::gdb::bool_t areAllNodeTypesAllowed](#)
True if all the node types are allowed.

5.47.1 Detailed Description

[SinglePairShortestPath](#) class.

Classes implementing this abstract class solve the shortest path problem in a graph from a given source node and to a given destination node.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.47.2 Constructor & Destructor Documentation

5.47.2.1 `SinglePairShortestPath::SinglePairShortestPath (sparksee::gdb::Session & s, sparksee::gdb::oid_t src, sparksee::gdb::oid_t dst) [protected]`

Creates a new instance.

Parameters:

s [in] [Session](#) to get the graph from and perform traversal.

src [in] Source node.

dst [dst] Destination node.

5.47.3 Member Function Documentation

5.47.3.1 `virtual sparksee::gdb::OIDList* SinglePairShortestPath::GetPathAsNodes () [pure virtual]`

Gets the shortest path between the source node and the destination node as an ordered set of nodes.

Returns:

Ordered set of node identifiers.

Implemented in [SinglePairShortestPathBFS](#), and [SinglePairShortestPathDijkstra](#).

5.47.3.2 `virtual sparksee::gdb::OIDList* SinglePairShortestPath::GetPathAsEdges () [pure virtual]`

Gets the shortest path between the source node and the destination node as an ordered set of edges.

Returns:

Ordered set of edge identifiers.

Implemented in [SinglePairShortestPathBFS](#), and [SinglePairShortestPathDijkstra](#).

5.47.3.3 `virtual sparksee::gdb::double64_t SinglePairShortestPath::GetCost () [pure virtual]`

Gets the cost of the shortest path.

The cost for unweighted algorithms is the number of hops of the shortest path. For weighted algorithms, the cost is the sum of the costs of the edges of the shortest path.

Returns:

The cost of the shortest path.

Implemented in [SinglePairShortestPathBFS](#), and [SinglePairShortestPathDijkstra](#).

5.47.3.4 `void ShortestPath::SetMaximumHops (sparksee::gdb::int32_t maxhops) [inherited]`

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

Parameters:

m [in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

5.47.3.5 virtual void ShortestPath::AddEdgeType (sparksee::gdb::type_t *t*, sparksee::gdb::EdgesDirection *d*) [virtual, inherited]

Allows for traversing edges of the given type.

Parameters:

t [in] Edge type.

d [in] Edge direction.

5.47.3.6 virtual void ShortestPath::AddAllEdgeTypes (sparksee::gdb::EdgesDirection *d*) [virtual, inherited]

Allows for traversing all edge types of the graph.

Parameters:

d [in] Edge direction.

5.47.3.7 virtual void ShortestPath::ExcludeNodes (sparksee::gdb::Objects & *nodes*) [virtual, inherited]

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters:

nodes [in] A set of node identifiers that must be kept intact until the destruction of the class.

5.47.3.8 virtual void ShortestPath::ExcludeEdges (sparksee::gdb::Objects & *edges*) [virtual, inherited]

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters:

edges [in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.47.3.9 virtual void ShortestPath::Run () [pure virtual, inherited]

Runs the algorithm.

This method can only be called once.

Implemented in [SinglePairShortestPathBFS](#), and [SinglePairShortestPathDijkstra](#).

5.47.4 Member Data Documentation

5.47.4.1 `sparksee::gdb::OIDList* SinglePairShortestPath::pathAsEdges` [protected]

Ordered set of edge identifiers representing the shortest path.

May be computed lazily when requested from the `pathAsNodes`.

5.47.4.2 `sparksee::gdb::int32_t ShortestPath::maxHops` [protected, inherited]

Maximum hops restriction.

It must be positive or zero. Zero means unlimited.

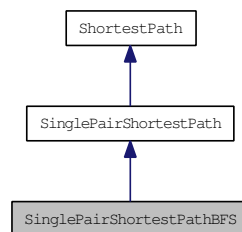
The documentation for this class was generated from the following file:

- `SinglePairShortestPath.h`

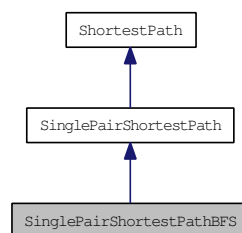
5.48 SinglePairShortestPathBFS Class Reference

[SinglePairShortestPathBFS](#) class.

Inheritance diagram for `SinglePairShortestPathBFS`:



Collaboration diagram for `SinglePairShortestPathBFS`:



Public Member Functions

- virtual `~SinglePairShortestPathBFS ()`
Destructor.
- virtual void `Run ()`
Executes the algorithm.

- virtual sparksee::gdb::OIDList * [GetPathAsNodes](#) ()
Gets the shortest path between the source node and the destination node as an ordered set of nodes.
- virtual sparksee::gdb::OIDList * [GetPathAsEdges](#) ()
Gets the shortest path between the source node and the destination node as an ordered set of edges.
- virtual [sparksee::gdb::double64_t](#) [GetCost](#) ()
Gets the cost of the shortest path.
- [SinglePairShortestPathBFS](#) (sparksee::gdb::Session &s, [sparksee::gdb::oid_t](#) src, [sparksee::gdb::oid_t](#) dst)
Creates a new instance.
- virtual void [CheckOnlyExistence](#) ()
Set that only the path existence must be calculated and not the path itself.
- virtual [sparksee::gdb::bool_t](#) [Exists](#) ()
Returns TRUE If a path exists or FALSE otherwise.
- void [SetMaximumHops](#) (sparksee::gdb::int32_t maxhops)
Sets the maximum hops restriction.
- virtual void [AddEdgeType](#) (sparksee::gdb::type_t t, sparksee::gdb::EdgesDirection d)
Allows for traversing edges of the given type.
- virtual void [AddAllEdgeTypes](#) (sparksee::gdb::EdgesDirection d)
Allows for traversing all edge types of the graph.
- virtual void [AddNodeType](#) (sparksee::gdb::type_t t)
Allows for traversing nodes of the given type.
- virtual void [AddAllNodeTypes](#) ()
Allows for traversing all node types of the graph.
- virtual void [ExcludeNodes](#) (sparksee::gdb::Objects &nodes)
Set which nodes can't be used.
- virtual void [ExcludeEdges](#) (sparksee::gdb::Objects &edges)
Set which edges can't be used.

Protected Member Functions

- void [AssertEdgeType](#) (sparksee::gdb::type_t edgetype)
Check that the given edge type is valid.
- void [AssertAddedEdges](#) ()
Check that edges had been added.
- void [AssertAddedNodes](#) ()

Check that nodes had been added.

- void [AssertNodeType](#) (sparksee::gdb::type_t nodetype)
Check that the given node type is valid.
- sparksee::gdb::bool_t [IsNodeTypeAllowed](#) (sparksee::gdb::oid_t nodeId)
Check if the given node has an allowed type.
- void [AssertNotComputed](#) ()
Check that the shortest path had not been calculated yet.
- void [AssertComputed](#) ()
Check that the shortest path had been calculated.
- sparksee::gdb::bool_t [IsNodeExcluded](#) (sparksee::gdb::oid_t node)
Check if the given node is forbidden.
- sparksee::gdb::bool_t [IsEdgeExcluded](#) (sparksee::gdb::oid_t edge)
Check if the given edge is forbidden.

Protected Attributes

- sparksee::gdb::oid_t [source](#)
Source node.
- sparksee::gdb::oid_t [destination](#)
Destination node.
- sparksee::gdb::OIDList * [pathAsNodes](#)
Ordered set of node identifiers representing the shortest path.
- sparksee::gdb::OIDList * [pathAsEdges](#)
Ordered set of edge identifiers representing the shortest path.
- sparksee::gdb::Session * [sess](#)
Session.
- sparksee::gdb::Graph * [graph](#)
Graph.
- std::map< [sparksee::gdb::type_t](#), [sparksee::gdb::EdgesDirection](#) > [edgeTypes](#)
Allowed edge types.
- std::vector< [sparksee::gdb::type_t](#) > [nodeTypes](#)
Allowed node types.
- sparksee::gdb::int32_t [maxHops](#)
Maximum hops restriction.

- `sparksee::gdb::bool_t` `computed`
True if the shortest path has been calculated.
- `sparksee::gdb::Objects *` `excludedNodes`
The set of excluded nodes.
- `sparksee::gdb::Objects *` `excludedEdges`
The set of excluded edges.
- `sparksee::gdb::bool_t` `areAllNodeTypesAllowed`
True if all the node types are allowed.

5.48.1 Detailed Description

`SinglePairShortestPathBFS` class.

It solves the single-pair shortest path problem using a BFS-based implementation.

It is a unweighted algorithm, that is it assumes all edges have the same cost.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.48.2 Constructor & Destructor Documentation

5.48.2.1 `SinglePairShortestPathBFS::SinglePairShortestPathBFS (sparksee::gdb::Session & s, sparksee::gdb::oid_t src, sparksee::gdb::oid_t dst)`

Creates a new instance.

Parameters:

- `s` [in] `Session` to get the graph from and perform traversal.
- `src` [in] Source node.
- `dst` [dst] Destination node.

5.48.3 Member Function Documentation

5.48.3.1 `virtual sparksee::gdb::OIDList* SinglePairShortestPathBFS::GetPathAsNodes ()` [virtual]

Gets the shortest path between the source node and the destination node as an ordered set of nodes.

Returns:

Ordered set of node identifiers.

Implements `SinglePairShortestPath`.

5.48.3.2 `virtual sparksee::gdb::OIDList* SinglePairShortestPathBFS::GetPathAsEdges ()` [virtual]

Gets the shortest path between the source node and the destination node as an ordered set of edges.

Returns:

Ordered set of edge identifiers.

Implements [SinglePairShortestPath](#).

5.48.3.3 `virtual sparksee::gdb::double64_t SinglePairShortestPathBFS::GetCost ()` [virtual]

Gets the cost of the shortest path.

The cost is the number of hops of the shortest path.

Returns:

The cost of the shortest path.

Implements [SinglePairShortestPath](#).

5.48.3.4 `virtual void SinglePairShortestPathBFS::CheckOnlyExistence ()` [virtual]

Set that only the path existence must be calculated and not the path itself.

That method should improve the performance of the algorithm, but a call to `GetPathAsNodes` or `GetPathAsEdges` will generate an exception even if the path exists.

5.48.3.5 `void ShortestPath::SetMaximumHops (sparksee::gdb::int32_t maxhops)` [inherited]

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

Parameters:

m [in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

5.48.3.6 `virtual void ShortestPath::AddEdgeType (sparksee::gdb::type_t t, sparksee::gdb::EdgesDirection d)` [virtual, inherited]

Allows for traversing edges of the given type.

Parameters:

t [in] Edge type.

d [in] Edge direction.

5.48.3.7 `virtual void ShortestPath::AddAllEdgeTypes (sparksee::gdb::EdgesDirection d)` [virtual, inherited]

Allows for traversing all edge types of the graph.

Parameters:

d [in] Edge direction.

5.48.3.8 `virtual void ShortestPath::ExcludeNodes (sparksee::gdb::Objects & nodes)` [virtual, inherited]

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters:

nodes [in] A set of node identifiers that must be kept intact until the destruction of the class.

5.48.3.9 `virtual void ShortestPath::ExcludeEdges (sparksee::gdb::Objects & edges)` [virtual, inherited]

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters:

edges [in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.48.4 Member Data Documentation

5.48.4.1 `sparksee::gdb::OIDList*` `SinglePairShortestPath::pathAsEdges` [protected, inherited]

Ordered set of edge identifiers representing the shortest path.

May be computed lazily when requested from the pathAsNodes.

5.48.4.2 `sparksee::gdb::int32_t` `ShortestPath::maxHops` [protected, inherited]

Maximum hops restriction.

It must be positive or zero. Zero means unlimited.

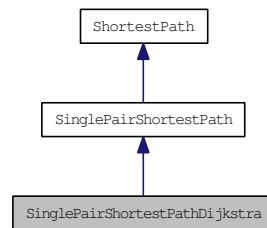
The documentation for this class was generated from the following file:

- SinglePairShortestPathBFS.h

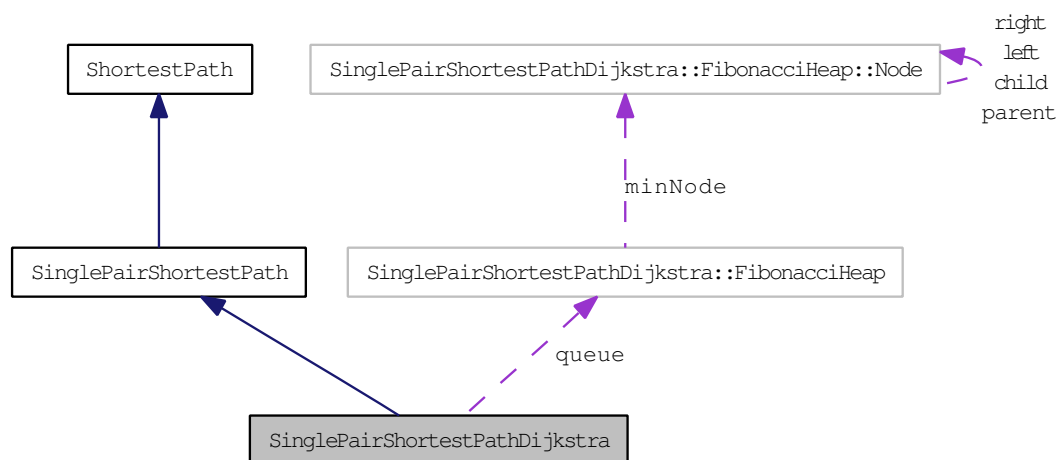
5.49 SinglePairShortestPathDijkstra Class Reference

[SinglePairShortestPathDijkstra](#) class.

Inheritance diagram for SinglePairShortestPathDijkstra:



Collaboration diagram for SinglePairShortestPathDijkstra:



Public Member Functions

- virtual `~SinglePairShortestPathDijkstra ()`
Destructor.
- virtual void `Run ()`
Executes the algorithm.
- virtual `sparksee::gdb::OIDList * GetPathAsNodes ()`
Gets the shortest path between the source node and the destination node as an ordered set of nodes.
- virtual `sparksee::gdb::OIDList * GetPathAsEdges ()`
Gets the shortest path between the source node and the destination node as an ordered set of edges.
- virtual `sparksee::gdb::double64_t GetCost ()`
Gets the cost of the shortest path.
- `SinglePairShortestPathDijkstra (sparksee::gdb::Session &s, sparksee::gdb::oid_t src, sparksee::gdb::oid_t dst)`
Creates a new instance.

- virtual void [AddWeightedEdgeType](#) (sparksee::gdb::type_t t, sparksee::gdb::EdgesDirection d, sparksee::gdb::attr_t attr)
Allows for traversing edges of the given type using the given attribute as the weight.
- virtual void [SetUnweightedEdgeCost](#) (sparksee::gdb::double64_t weight)
Sets the weight assigned to the unweighted edges.
- virtual sparksee::gdb::bool_t [Exists](#) ()
Returns TRUE If a path exists or FALSE otherwise.
- void [SetMaximumHops](#) (sparksee::gdb::int32_t maxhops)
Sets the maximum hops restriction.
- virtual void [AddEdgeType](#) (sparksee::gdb::type_t t, sparksee::gdb::EdgesDirection d)
Allows for traversing edges of the given type.
- virtual void [AddAllEdgeTypes](#) (sparksee::gdb::EdgesDirection d)
Allows for traversing all edge types of the graph.
- virtual void [AddNodeType](#) (sparksee::gdb::type_t t)
Allows for traversing nodes of the given type.
- virtual void [AddAllNodeTypes](#) ()
Allows for traversing all node types of the graph.
- virtual void [ExcludeNodes](#) (sparksee::gdb::Objects &nodes)
Set which nodes can't be used.
- virtual void [ExcludeEdges](#) (sparksee::gdb::Objects &edges)
Set which edges can't be used.

Protected Member Functions

- void [AssertEdgeType](#) (sparksee::gdb::type_t edgetype)
Check that the given edge type is valid.
- void [AssertAddedEdges](#) ()
Check that edges had been added.
- void [AssertAddedNodes](#) ()
Check that nodes had been added.
- void [AssertNodeType](#) (sparksee::gdb::type_t nodetype)
Check that the given node type is valid.
- sparksee::gdb::bool_t [IsNodeTypeAllowed](#) (sparksee::gdb::oid_t nodeId)
Check if the given node has an allowed type.
- void [AssertNotComputed](#) ()

Check that the shortest path had not been calculated yet.

- void [AssertComputed](#) ()
Check that the shortest path had been calculated.
- [sparksee::gdb::bool_t IsNodeExcluded](#) ([sparksee::gdb::oid_t](#) node)
Check if the given node is forbidden.
- [sparksee::gdb::bool_t IsEdgeExcluded](#) ([sparksee::gdb::oid_t](#) edge)
Check if the given edge is forbidden.

Protected Attributes

- [sparksee::gdb::oid_t](#) source
Source node.
- [sparksee::gdb::oid_t](#) destination
Destination node.
- [sparksee::gdb::OIDList](#) * [pathAsNodes](#)
Ordered set of node identifiers representing the shortest path.
- [sparksee::gdb::OIDList](#) * [pathAsEdges](#)
Ordered set of edge identifiers representing the shortest path.
- [sparksee::gdb::Session](#) * [sess](#)
Session.
- [sparksee::gdb::Graph](#) * [graph](#)
Graph.
- [std::map](#)< [sparksee::gdb::type_t](#), [sparksee::gdb::EdgesDirection](#) > [edgeTypes](#)
Allowed edge types.
- [std::vector](#)< [sparksee::gdb::type_t](#) > [nodeTypes](#)
Allowed node types.
- [sparksee::gdb::int32_t](#) maxHops
Maximum hops restriction.
- [sparksee::gdb::bool_t](#) computed
True if the shortest path has been calculated.
- [sparksee::gdb::Objects](#) * [excludedNodes](#)
The set of excluded nodes.
- [sparksee::gdb::Objects](#) * [excludedEdges](#)
The set of excluded edges.

- `sparksee::gdb::bool_t areAllNodeTypesAllowed`

True if all the node types are allowed.

Classes

- class **FibonacciHeap**

5.49.1 Detailed Description

`SinglePairShortestPathDijkstra` class.

It solves the single-pair shortest path problem using a Dijkstra-based implementation.

It is a weighted algorithm, so it takes into account the cost of the edges to compute a minimum-cost shortest path. That is, the user may set for each edge type which attribute should be used to retrieve the cost of the edge. If no attribute is given for an edge type, this will assume the edge has a fixed cost (the default is 1). Only numerical attribute can be set as weight attributes (that is Long, Integer or Double attributes are allowed).

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.49.2 Constructor & Destructor Documentation

5.49.2.1 `SinglePairShortestPathDijkstra::SinglePairShortestPathDijkstra (sparksee::gdb::Session & s, sparksee::gdb::oid_t src, sparksee::gdb::oid_t dst)`

Creates a new instance.

Parameters:

- `s` [in] [Session](#) to get the graph from and perform traversal.
- `src` [in] Source node.
- `dst` [dst] Destination node.

5.49.3 Member Function Documentation

5.49.3.1 `virtual sparksee::gdb::OIDList* SinglePairShortestPathDijkstra::GetPathAsNodes ()` [virtual]

Gets the shortest path between the source node and the destination node as an ordered set of nodes.

Returns:

Ordered set of node identifiers.

Implements [SinglePairShortestPath](#).

5.49.3.2 `virtual sparksee::gdb::OIDList* SinglePairShortestPathDijkstra::GetPathAsEdges ()`
[virtual]

Gets the shortest path between the source node and the destination node as an ordered set of edges.

Returns:

Ordered set of edge identifiers.

Implements [SinglePairShortestPath](#).

5.49.3.3 `virtual sparksee::gdb::double64_t SinglePairShortestPathDijkstra::GetCost ()`
[virtual]

Gets the cost of the shortest path.

The cost is the sum of the weights of the edges in the shortest path.

Returns:

The cost of the shortest path.

Implements [SinglePairShortestPath](#).

5.49.3.4 `virtual void SinglePairShortestPathDijkstra::AddWeightedEdgeType (sparksee::gdb::type_t t, sparksee::gdb::EdgesDirection d, sparksee::gdb::attr_t attr)`
[virtual]

Allows for traversing edges of the given type using the given attribute as the weight.

Parameters:

t [in] Edge type.

d [in] Edge direction.

attr [in] [Attribute](#) to be used as the weight. It must be a global attribute or an attribute of the given edge type.

5.49.3.5 `virtual void SinglePairShortestPathDijkstra::SetUnweightedEdgeCost (sparksee::gdb::double64_t weight)` [virtual]

Sets the weight assigned to the unweighted edges.

All the edges from the types added without an explicit weight attribute will get this weight. The default weight for this edges is 1.

5.49.3.6 `void ShortestPath::SetMaximumHops (sparksee::gdb::int32_t maxhops)` [inherited]

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

Parameters:

m [in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

5.49.3.7 `virtual void ShortestPath::AddEdgeType (sparksee::gdb::type_t t, sparksee::gdb::EdgesDirection d)` [virtual, inherited]

Allows for traversing edges of the given type.

Parameters:

- t* [in] Edge type.
- d* [in] Edge direction.

5.49.3.8 `virtual void ShortestPath::AddAllEdgeTypes (sparksee::gdb::EdgesDirection d)` [virtual, inherited]

Allows for traversing all edge types of the graph.

Parameters:

- d* [in] Edge direction.

5.49.3.9 `virtual void ShortestPath::ExcludeNodes (sparksee::gdb::Objects & nodes)` [virtual, inherited]

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters:

- nodes* [in] A set of node identifiers that must be kept intact until the destruction of the class.

5.49.3.10 `virtual void ShortestPath::ExcludeEdges (sparksee::gdb::Objects & edges)` [virtual, inherited]

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters:

- edges* [in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.49.4 Member Data Documentation

5.49.4.1 `sparksee::gdb::OIDList*` `SinglePairShortestPath::pathAsEdges` [protected, inherited]

Ordered set of edge identifiers representing the shortest path.

May be computed lazily when requested from the pathAsNodes.

5.49.4.2 sparksee::gdb::int32_t ShortestPath::maxHops [protected, inherited]

Maximum hops restriction.

It must be positive or zero. Zero means unlimited.

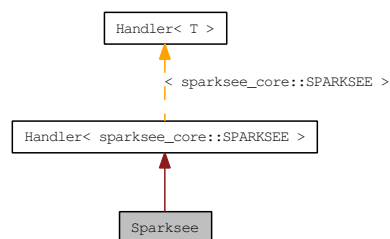
The documentation for this class was generated from the following file:

- SinglePairShortestPathDijkstra.h

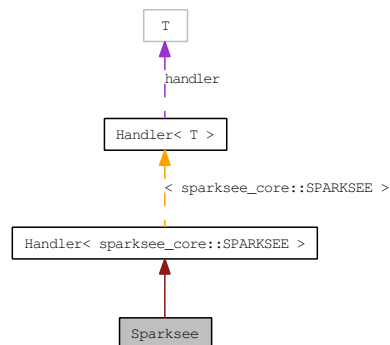
5.50 Sparksee Class Reference

[Sparksee](#) class.

Inheritance diagram for Sparksee:



Collaboration diagram for Sparksee:



Public Member Functions

- [Sparksee](#) (const [SparkseeConfig](#) &config)
Creates a new instance.
- virtual [~Sparksee](#) ()
Destructor.
- [Database](#) * [Create](#) (const std::wstring &path, const std::wstring &alias) throw (sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)
Creates a new [Database](#) instance.

- `Database * Open (const std::wstring &path, bool_t read) throw (sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)`
Opens an existing [Database](#) instance.
- `Database * Restore (const std::wstring &path, const std::wstring &backupFile) throw (sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)`
Restores a [Database](#) from a backup file.

Static Public Attributes

- static const std::wstring [Version](#)
Sparksee version.

5.50.1 Detailed Description

[Sparksee](#) class.

All [Sparksee](#) programs must have one single [Sparksee](#) instance to manage one or more [Database](#) instances. This class allows for the creation of new Databases or open an existing one.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.50.2 Constructor & Destructor Documentation

5.50.2.1 Sparksee::Sparksee (const SparkseeConfig & config)

Creates a new instance.

Parameters:

config [in] [Sparksee](#) configuration.

5.50.3 Member Function Documentation

5.50.3.1 Database* Sparksee::Create (const std::wstring & path, const std::wstring & alias) throw (sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)

Creates a new [Database](#) instance.

Parameters:

path [in] [Database](#) storage file.

alias [in] [Database](#) alias name.

Returns:

A [Database](#) instance.

Exceptions:

[FileNotFoundException](#) If the given file cannot be created.

5.50.3.2 Database* Sparksee::Open (const std::wstring & *path*, bool_t *read*) throw (sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)

Opens an existing [Database](#) instance.

Parameters:

- path* [in] [Database](#) storage file.
- read* [in] If TRUE, open [Database](#) in read-only mode.

Returns:

A [Database](#) instance.

Exceptions:

[FileNotFoundException](#) If the given file does not exist.

5.50.3.3 Database* Sparksee::Restore (const std::wstring & *path*, const std::wstring & *backupFile*) throw (sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)

Restores a [Database](#) from a backup file.

See the [Graph](#) class Backup method.

Parameters:

- path* [in] [Database](#) storage file.
- backupFile* [in] The Backup file to be restored.

Returns:

A [Database](#) instance.

Exceptions:

[FileNotFoundException](#) If the given file cannot be created, or the exported data file does not exists.

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

- [Sparksee.h](#)

5.51 SparkseeConfig Class Reference

[Sparksee](#) configuration class.

Public Member Functions

- [SparkseeConfig](#) ()
Creates a new instance.
- virtual [~SparkseeConfig](#) ()
Destructor.

- `int32_t GetExtentSize () const`
Gets the size of a extent.
- `void SetExtentSize (int32_t v)`
Sets the size of a pool frame in number of extents.
- `int32_t GetExtentPages () const`
Gets the number of pages per extent.
- `void SetExtentPages (int32_t v)`
Sets the number of pages per extent.
- `int32_t GetPoolFrameSize () const`
Gets the size of a pool frame in number of extents.
- `void SetPoolFrameSize (int32_t v)`
Sets the size of a pool frame in number of extents.
- `int32_t GetPoolPersistentMinSize () const`
Gets the minimum size for the persistent pool in number of frames.
- `void SetPoolPersistentMinSize (int32_t v)`
Sets the minimum size for the persistent pool in number of frames.
- `int32_t GetPoolPersistentMaxSize () const`
Gets the maximum size for the persistent pool in number of frames.
- `void SetPoolPersistentMaxSize (int32_t v)`
Sets the maximum size for the persistent pool in number of frames.
- `int32_t GetPoolTemporaryMinSize () const`
Gets the minimum size for the temporary pool in number of frames.
- `void SetPoolTemporaryMinSize (int32_t v)`
Sets the minimum size for the temporary pool in number of frames.
- `int32_t GetPoolTemporaryMaxSize () const`
Gets the maximum size for the temporary pool in number of frames.
- `void SetPoolTemporaryMaxSize (int32_t v)`
Sets the maximum size for the temporary pool in number of frames.
- `int32_t GetCacheMaxSize () const`
Gets the maximum size for the cache (all pools) in MB.
- `void SetCacheMaxSize (int32_t v)`
Sets the maximum size for the cache (all pools) in MB.
- `const std::wstring & GetLicense () const`
Gets the license code.

- void [SetLicense](#) (const std::wstring &v)
Sets the license code.
- const std::wstring & [GetLogFile](#) () const
Gets the log file.
- void [SetLogFile](#) (const std::wstring &v)
Sets the log file.
- [LogLevel](#) [GetLogLevel](#) () const
Gets the log level.
- void [SetLogLevel](#) ([LogLevel](#) v)
Sets the log level.
- [bool_t](#) [GetCacheStatisticsEnabled](#) () const
Gets whether cache statistics are enabled or disabled.
- void [SetCacheStatisticsEnabled](#) ([bool_t](#) v)
Enables or disables cache statistics.
- const std::wstring & [GetCacheStatisticsFile](#) () const
Gets the cache statistics log file.
- void [SetCacheStatisticsFile](#) (const std::wstring &v)
Sets the cache statistics log file.
- [int64_t](#) [GetCacheStatisticsSnapshotTime](#) () const
Gets the cache statistics snapshot time in microseconds.
- void [SetCacheStatisticsSnapshotTime](#) ([int64_t](#) v)
Sets the cache statistics snapshot time.
- [bool_t](#) [GetRecoveryEnabled](#) () const
Gets whether the recovery is enabled or disabled.
- void [SetRecoveryEnabled](#) ([bool_t](#) v)
Enables or disables the recovery.
- const std::wstring & [GetRecoveryLogFile](#) () const
Gets the recovery log file.
- void [SetRecoveryLogFile](#) (const std::wstring &v)
Sets the recovery log file.
- [int32_t](#) [GetRecoveryCacheMaxSize](#) () const
Gets the maximum size for the recovery log cache in extents.
- void [SetRecoveryCacheMaxSize](#) ([int32_t](#) v)

Sets the maximum size for the recovery log cache in extents.

- [int64_t GetRecoveryCheckpointTime \(\)](#) const
Gets the delay time (in microseconds) between automatic checkpoints.
- void [SetRecoveryCheckpointTime \(int64_t v\)](#)
Sets the delay time (in microseconds) between automatic checkpoints.
- [bool_t GetHighAvailabilityEnabled \(\)](#) const
Gets whether high availability mode is enabled or disabled.
- void [SetHighAvailabilityEnabled \(bool_t v\)](#)
Enables or disables high availability mode.
- const std::wstring & [GetHighAvailabilityIP \(\)](#) const
Gets the IP address and port of the instance.
- void [SetHighAvailabilityIP \(const std::wstring &v\)](#)
Sets the IP address and port of the instance.
- const std::wstring & [GetHighAvailabilityCoordinators \(\)](#) const
Gets the coordinators address and port list.
- void [SetHighAvailabilityCoordinators \(const std::wstring &v\)](#)
Sets the coordinators address and port list.
- [int64_t GetHighAvailabilitySynchronization \(\)](#) const
Gets the synchronization polling time.
- void [SetHighAvailabilitySynchronization \(int64_t v\)](#)
Sets the synchronization polling time.
- [int64_t GetHighAvailabilityMasterHistory \(\)](#) const
Gets the master's history log.
- void [SetHighAvailabilityMasterHistory \(int64_t v\)](#)
Sets the master's history log.

5.51.1 Detailed Description

[Sparksee](#) configuration class.

If not specified, 0 means unlimited which is the maximum available. For the pools that's the total cache size. For the cache unlimited means nearly all the physical memory of the computer.

For each field, there is a default value. This value can be overridden with values from a properties file (see [SparkseeProperties](#) class). Also, this settings can be overridden calling a specific setter.

For each field, it is shown its default value and the property to override this value:

Extent size: 4KB ('sparksee.storage.extentsize' at [SparkseeProperties](#)).

Pages per extent: 1 page ('sparksee.storage.extentpages' at [SparkseeProperties](#)).

Pool frame size: 1 extent ('sparksee.io.pool.frame.size' at [SparkseeProperties](#)).

Minimum size for the persistent pool: 64 frames ('sparksee.io.pool.persistent.minsize' at [SparkseeProperties](#)).

Maximum size for the persistent pool: 0 frames ('sparksee.io.pool.persistent.maxsize' at [SparkseeProperties](#)).

Minimum size for the temporary pool: 16 frames ('sparksee.io.pool.temporal.minsize' at [SparkseeProperties](#)).

Maximum size for the temporary pool: 0 frames ('sparksee.io.pool.temporal.maxsize' at [SparkseeProperties](#)).

Maximum size for the cache (all pools): 0 MB ('sparksee.io.cache.maxsize' at [SparkseeProperties](#)).

License code: "" ('sparksee.license' at [SparkseeProperties](#)). No license code means evaluation license.

Log level: Info ('sparksee.log.level' at [SparkseeProperties](#)).

Log file: "sparksee.log" ('sparksee.log.file' at [SparkseeProperties](#)).

Cache statistics: false (disabled) ('sparksee.cache.statistics' at [SparkseeProperties](#)).

Cache statistics log file: "statistics.log" ('sparksee.cache.statisticsFile' at [SparkseeProperties](#)).

Cache statistics snapshot time: 1000 msecs [TimeUnit] ('sparksee.cache.statisticsSnapshotTime' at [SparkseeProperties](#)).

Recovery enabled: false ('sparksee.io.recovery' at [SparkseeProperties](#)).

Recovery log file: "" ('sparksee.io.recovery.logfile' at [SparkseeProperties](#)).

Recovery cache max size: 1MB ('sparksee.io.recovery.cachesize' at [SparkseeProperties](#)).

Recovery checkpoint time: 60 seconds [TimeUnit] ('sparksee.io.recovery.checkpointTime' at [SparkseeProperties](#)).

High-availability: false (disabled) ('sparksee.ha' at [SparkseeProperties](#)).

High-availability coordinators: "" ('sparksee.ha.coordinators' at [SparkseeProperties](#)).

High-availability IP: "" ('sparksee.ha.ip' at [SparkseeProperties](#)).

High-availability sync polling: 0 (disabled) [TimeUnit] ('sparksee.ha.sync' at [SparkseeProperties](#)).

High-availability master history: 1D (1 day) [TimeUnit] ('sparksee.ha.master.history' at [SparkseeProperties](#)).

Use of TimeUnit:

Those variables using TimeUnit allow for:

<X>[D|H|M|S|s|m|u]

where <X> is a number followed by an optional character which represents the unit: D for days, H for hours, M for minutes, S or s for seconds, m for milliseconds and u for microseconds. If no unit character is given, seconds are assumed.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.51.2 Constructor & Destructor Documentation

5.51.2.1 SparkseeConfig::SparkseeConfig ()

Creates a new instance.

Values are set with default values.

5.51.3 Member Function Documentation

5.51.3.1 int32_t SparkseeConfig::GetExtentSize () const [inline]

Gets the size of a extent.

Returns:

The size of a extent in KB.

5.51.3.2 void SparkseeConfig::SetExtentSize (int32_t v) [inline]

Sets the size of a pool frame in number of extents.

Parameters:

v [in] The size of a extent size in KB. An extent can have a size between 4KB and 64KB, and it must be a power of 2.

5.51.3.3 int32_t SparkseeConfig::GetExtentPages () const [inline]

Gets the number of pages per extent.

Returns:

The number of pages per extent.

5.51.3.4 void SparkseeConfig::SetExtentPages (int32_t v) [inline]

Sets the number of pages per extent.

Parameters:

v [in] The number of pages. It must be at least 1 page and the page size must be greater than or equal to 4KB.

5.51.3.5 int32_t SparkseeConfig::GetPoolFrameSize () const [inline]

Gets the size of a pool frame in number of extents.

Returns:

The size of a pool frame in number of extents.

5.51.3.6 void SparkseeConfig::SetPoolFrameSize (int32_t v) [inline]

Sets the size of a pool frame in number of extents.

Parameters:

v [in] The size of a pool frame in number of extents. It must be non-negative.

5.51.3.7 int32_t SparkseeConfig::GetPoolPersistentMinSize () const [inline]

Gets the minimum size for the persistent pool in number of frames.

Returns:

The minimum size for the persistent pool in number of frames.

5.51.3.8 void SparkseeConfig::SetPoolPersistentMinSize (int32_t v) [inline]

Sets the minimum size for the persistent pool in number of frames.

Parameters:

v [in] The minimum size for the persistent pool in number of frames. It must be non-negative.

5.51.3.9 int32_t SparkseeConfig::GetPoolPersistentMaxSize () const [inline]

Gets the maximum size for the persistent pool in number of frames.

Returns:

The maximum size for the persistent pool in number of frames.

5.51.3.10 void SparkseeConfig::SetPoolPersistentMaxSize (int32_t v) [inline]

Sets the maximum size for the persistent pool in number of frames.

Parameters:

v [in] The maximum size for the persistent pool in number of frames. It must be non-negative.

5.51.3.11 int32_t SparkseeConfig::GetPoolTemporaryMinSize () const [inline]

Gets the minimum size for the temporary pool in number of frames.

Returns:

The minimum size for the temporary pool in number of frames.

5.51.3.12 void SparkseeConfig::SetPoolTemporaryMinSize (int32_t v) [inline]

Sets the minimum size for the temporary pool in number of frames.

Parameters:

v [in] The minimum size for the temporary pool in number of frames. It must be non-negative.

5.51.3.13 int32_t SparkseeConfig::GetPoolTemporaryMaxSize () const [inline]

Gets the maximum size for the temporary pool in number of frames.

Returns:

The maximum size for the temporary pool in number of frames.

5.51.3.14 void SparkseeConfig::SetPoolTemporaryMaxSize (int32_t v) [inline]

Sets the maximum size for the temporary pool in number of frames.

Parameters:

v [in] The maximum size for the temporary pool in number of frames. It must be non-negative.

5.51.3.15 int32_t SparkseeConfig::GetCacheMaxSize () const [inline]

Gets the maximum size for the cache (all pools) in MB.

Returns:

The maximum size for the cache (all pools) in MB.

5.51.3.16 void SparkseeConfig::SetCacheMaxSize (int32_t v) [inline]

Sets the maximum size for the cache (all pools) in MB.

Parameters:

v [in] The maximum size for the cache (all pools) in MB. It must be non-negative.

5.51.3.17 const std::wstring& SparkseeConfig::GetLicense () const [inline]

Gets the license code.

Returns:

The license code.

5.51.3.18 void SparkseeConfig::SetLicense (const std::wstring & v)

Sets the license code.

Parameters:

v [in] The license code.

5.51.3.19 `const std::wstring& SparkseeConfig::GetLogFile () const` `[inline]`

Gets the log file.

Returns:

The log file.

5.51.3.20 `void SparkseeConfig::SetLogFile (const std::wstring & v)` `[inline]`

Sets the log file.

Parameters:

v [in] The log file.

5.51.3.21 `LogLevel SparkseeConfig::GetLogLevel () const` `[inline]`

Gets the log level.

Returns:

The LogLevel.

5.51.3.22 `void SparkseeConfig::SetLogLevel (LogLevel v)` `[inline]`

Sets the log level.

Parameters:

v [in] The LogLevel.

5.51.3.23 `bool_t SparkseeConfig::GetCacheStatisticsEnabled () const` `[inline]`

Gets whether cache statistics are enabled or disabled.

Returns:

TRUE if cache statistics are enabled, FALSE otherwise.

5.51.3.24 `void SparkseeConfig::SetCacheStatisticsEnabled (bool_t v)` `[inline]`

Enables or disables cache statistics.

Parameters:

v [in] If TRUE this enables cache statistics, if FALSE this disables cache statistics.

5.51.3.25 `const std::wstring& SparkseeConfig::GetCacheStatisticsFile () const` `[inline]`

Gets the cache statistics log file.

Useless if cache statistics are disabled.

Returns:

The cache statistics log file.

5.51.3.26 `void SparkseeConfig::SetCacheStatisticsFile (const std::wstring & v)` `[inline]`

Sets the cache statistics log file.

Useless if cache statistics are disabled.

Parameters:

v [in] The cache statistics log file.

5.51.3.27 `int64_t SparkseeConfig::GetCacheStatisticsSnapshotTime () const` `[inline]`

Gets the cache statistics snapshot time in microseconds.

Useless if cache statistics are disabled.

Returns:

The cache statistics snapshot time in microseconds.

5.51.3.28 `void SparkseeConfig::SetCacheStatisticsSnapshotTime (int64_t v)` `[inline]`

Sets the cache statistics snapshot time.

Useless if cache statistics are disabled.

Parameters:

v [in] The cache statistics snapshot time in microseconds.

5.51.3.29 `bool_t SparkseeConfig::GetRecoveryEnabled () const` `[inline]`

Gets whether the recovery is enabled or disabled.

Returns:

TRUE if the recovery is enabled, FALSE otherwise.

5.51.3.30 `void SparkseeConfig::SetRecoveryEnabled (bool_t v)` `[inline]`

Enables or disables the recovery.

Parameters:

v [in] If TRUE this enables the recovery, if FALSE then disables it.

5.51.3.31 `const std::wstring& SparkseeConfig::GetRecoveryLogFile () const` [inline]

Gets the recovery log file.

Returns:

The recovery log file.

5.51.3.32 `void SparkseeConfig::SetRecoveryLogFile (const std::wstring & v)` [inline]

Sets the recovery log file.

Parameters:

v [in] The recovery log file. Left it empty for the default log file (same as <database_file_name>.log)

5.51.3.33 `int32_t SparkseeConfig::GetRecoveryCacheMaxSize () const` [inline]

Gets the maximum size for the recovery log cache in extents.

Returns:

The maximum size for the recovery log cache in extents.

5.51.3.34 `void SparkseeConfig::SetRecoveryCacheMaxSize (int32_t v)` [inline]

Sets the maximum size for the recovery log cache in extents.

Parameters:

v [in] The maximum size for the recovery log cache in extents. A 0 sets the default value (extents up to 1MB).

5.51.3.35 `int64_t SparkseeConfig::GetRecoveryCheckpointTime () const` [inline]

Gets the delay time (in microseconds) between automatic checkpoints.

Returns:

The delay time (in microseconds) between automatic checkpoints.

5.51.3.36 `void SparkseeConfig::SetRecoveryCheckpointTime (int64_t v)` [inline]

Sets the delay time (in microseconds) between automatic checkpoints.

Parameters:

v [in] The delay time (in microseconds) between automatic checkpoints. A 0 forces a checkpoint after each committed transaction.

5.51.3.37 bool_t SparkseeConfig::GetHighAvailabilityEnabled () const [inline]

Gets whether high availability mode is enabled or disabled.

Returns:

TRUE if high availability mode is enabled, FALSE otherwise.

5.51.3.38 void SparkseeConfig::SetHighAvailabilityEnabled (bool_t v) [inline]

Enables or disables high availability mode.

Parameters:

v [in] If TRUE this enables high availability mode, if FALSE this disables high availability mode.

5.51.3.39 const std::wstring& SparkseeConfig::GetHighAvailabilityIP () const [inline]

Gets the IP address and port of the instance.

Returns:

The IP address and port of the instance.

5.51.3.40 void SparkseeConfig::SetHighAvailabilityIP (const std::wstring & v) [inline]

Sets the IP address and port of the instance.

Parameters:

v [in] The IP address and port of the instance.

5.51.3.41 const std::wstring& SparkseeConfig::GetHighAvailabilityCoordinators () const [inline]

Gets the coordinators address and port list.

Returns:

The coordinators address and port list.

5.51.3.42 void SparkseeConfig::SetHighAvailabilityCoordinators (const std::wstring & v) [inline]

Sets the coordinators address and port list.

Parameters:

v [in] The coordinators address and port list.

5.51.3.43 int64_t SparkseeConfig::GetHighAvailabilitySynchronization () const [inline]

Gets the synchronization polling time.

Returns:

The Synchronization polling time.

5.51.3.44 void SparkseeConfig::SetHighAvailabilitySynchronization (int64_t v) [inline]

Sets the synchronization polling time.

Parameters:

v [in] The synchronization polling time.

5.51.3.45 int64_t SparkseeConfig::GetHighAvailabilityMasterHistory () const [inline]

Gets the master's history log.

Returns:

The master's history log.

5.51.3.46 void SparkseeConfig::SetHighAvailabilityMasterHistory (int64_t v) [inline]

Sets the master's history log.

Parameters:

v [in] The master's history log.

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

- Sparksee.h

5.52 SparkseeProperties Class Reference

[Sparksee](#) properties file.

Static Public Member Functions

- static void [Load](#) (const std::wstring &path)
Loads properties from the given file path.
- static const std::wstring & [Get](#) (const std::wstring &key, const std::wstring &def)
Gets a property.
- static [int32_t](#) [GetInteger](#) (const std::wstring &key, [int32_t](#) def)
Gets a property as an integer.

- static `bool_t GetBoolean` (const std::wstring &key, `bool_t` def)
Gets a property as a boolean.
- static `int64_t GetTimeUnit` (const std::wstring &key, `int64_t` def)
Gets a property as a time unit.

5.52.1 Detailed Description

Sparksee properties file.

This class is implemented as a singleton, so all public methods are static.

It allows for getting the property values stored in a properties file. A properties file is a file where there is one line per property. A property is defined by a key and a value as follows: key=value

By default, this loads properties from the file './sparksee.cfg'. The user may choose to load a different file by calling the method `Load()`.

If the default properties file or the one loaded by the user do not exist, then this behaves as loading an empty properties file.

5.52.2 Member Function Documentation

5.52.2.1 static void SparkseeProperties::Load (const std::wstring & path) [static]

Loads properties from the given file path.

Parameters:

path [in] File path to load properties from.

5.52.2.2 static const std::wstring& SparkseeProperties::Get (const std::wstring & key, const std::wstring & def) [static]

Gets a property.

Parameters:

key [in] The name of the property to lookup.

def [in] Default value to be returned in case there is no property with the name key.

Returns:

The value of the property, or def if the key is not found.

5.52.2.3 static int32_t SparkseeProperties::GetInteger (const std::wstring & key, int32_t def) [static]

Gets a property as an integer.

Parameters:

key [in] The name of the property to lookup.

def [in] Default value to be returned in case there is no property with the name key.

Returns:

The property value, or *def* if the key is not found or in case of error.

5.52.2.4 `static bool_t SparkseeProperties::GetBoolean (const std::wstring & key, bool_t def)`
[static]

Gets a property as a boolean.

Parameters:

key [in] The name of the property to lookup.

def [in] Default value to be returned in case there is no property with the name key.

Returns:

The property value, or *def* if the key is not found or in case of error.

5.52.2.5 `static int64_t SparkseeProperties::GetTimeUnit (const std::wstring & key, int64_t def)`
[static]

Gets a property as a time unit.

A time unit is a string representation of a time duration with a time unit such as '10s' or '3H'.

Valid format for the string representation: Blanks at the begining or at the end are ignored. No blanks are allowed between the time duration and the unit time.

Allowed time units: 'D' for days, 'H' for hours, 'M' for minutes, 'S' o 's' for seconds, 'm' for milliseconds and 'u' for microseconds.

There is a special case: If no time unit is given, seconds is the default. So, '10' means 10 seconds.

Parameters:

key [in] The name of the property to lookup.

def [in] The default value (in microseconds) to be returned in case there is no property with the name key.

Returns:

The time duration in microseconds, or *def* if the key is not found or in case of error.

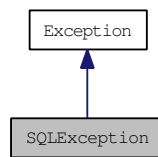
The documentation for this class was generated from the following file:

- Sparksee.h

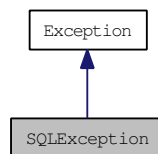
5.53 SQLException Class Reference

SQL exception class.

Inheritance diagram for SQLException:



Collaboration diagram for SQLException:



Public Member Functions

- [SQLException](#) ()
Creates a new instance.
- [SQLException](#) (const std::string &mess)
Creates a new instance.
- [SQLException](#) (const sparksee_core::SQLError *core_error)
Creates a new instance.
- virtual [~SQLException](#) ()
Destructor.
- const std::string & [Message](#) () const
Gets the message of the exception.
- void [SetMessage](#) (const std::string &mess)
Sets the message of the exception.

Protected Attributes

- std::string [message](#)
Message of the exception.

5.53.1 Detailed Description

SQL exception class.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.53.2 Constructor & Destructor Documentation

5.53.2.1 SQLException::SQLException (const std::string & *mess*)

Creates a new instance.

Parameters:

mess [in] Message of the exception.

5.53.2.2 SQLException::SQLException (const sparksee_core::SQLException * *core_error*)

Creates a new instance.

Parameters:

core_error [in] Sparkseecore SQLException.

5.53.3 Member Function Documentation

5.53.3.1 const std::string& Exception::Message () const [inherited]

Gets the message of the exception.

It should be called GetMessage but this is not possible because of a macro defined in Windows.

Returns:

The message of the exception.

5.53.3.2 void Exception::SetMessage (const std::string & *mess*) [inherited]

Sets the message of the exception.

Parameters:

mess [in] Message.

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

- Exception.h

5.54 StringList Class Reference

String list.

Public Member Functions

- [int32_t Count](#) () const
Number of elements in the list.
- [StringListIterator * Iterator](#) ()
Gets a new [StringListIterator](#).

- [StringList \(\)](#)
Constructor.
- [StringList \(const std::vector< std::wstring > &v\)](#)
Constructor.
- [~StringList \(\)](#)
Destructor.
- void [Add](#) (const std::wstring &str)
Adds a String at the end of the list.
- void [Clear](#) ()
Clears the list.

5.54.1 Detailed Description

String list.

It stores a String (unicode) list.

Use [StringListIterator](#) to access all elements into this collection.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.54.2 Constructor & Destructor Documentation

5.54.2.1 StringList::StringList ()

Constructor.

This creates an empty list.

5.54.2.2 StringList::StringList (const std::vector< std::wstring > &v)

Constructor.

Parameters:

v [in] Vector.

5.54.3 Member Function Documentation

5.54.3.1 int32_t StringList::Count () const [inline]

Number of elements in the list.

Returns:

Number of elements in the list.

5.54.3.2 StringListIterator* StringList::Iterator ()

Gets a new [StringListIterator](#).

Returns:

[StringListIterator](#) instance.

5.54.3.3 void StringList::Add (const std::wstring & *str*) [inline]

Adds a String at the end of the list.

Parameters:

str [in] String.

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

- [Graph_data.h](#)

5.55 StringListIterator Class Reference

[StringList](#) iterator class.

Public Member Functions

- [~StringListIterator](#) ()
Destructor.
- const std::wstring & [Next](#) ()
Moves to the next element.
- [bool_t](#) [HasNext](#) ()
Gets if there are more elements.

Friends

- class [StringList](#)

5.55.1 Detailed Description

[StringList](#) iterator class.

Iterator to traverse all the strings into a [StringList](#) instance.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.55.2 Member Function Documentation

5.55.2.1 `const std::wstring& StringListIterator::Next ()` [inline]

Moves to the next element.

Returns:

The next element.

5.55.2.2 `bool_t StringListIterator::HasNext ()` [inline]

Gets if there are more elements.

Returns:

TRUE if there are more elements, FALSE otherwise.

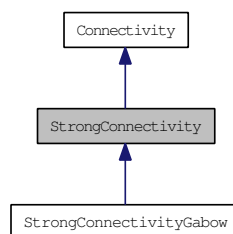
The documentation for this class was generated from the following file:

- Graph_data.h

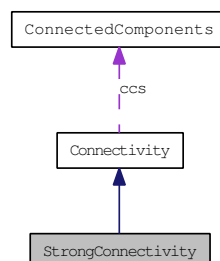
5.56 StrongConnectivity Class Reference

[StrongConnectivity](#) class.

Inheritance diagram for StrongConnectivity:



Collaboration diagram for StrongConnectivity:



Public Member Functions

- virtual `~StrongConnectivity ()`

Destructor.

- virtual void [AddEdgeType](#) (sparksee::gdb::type_t t, sparksee::gdb::EdgesDirection d)
Allows connectivity through edges of the given type.
- virtual void [AddAllEdgeTypes](#) (sparksee::gdb::EdgesDirection d)
Allows connectivity through all edge types of the graph.
- virtual void [AddNodeType](#) (sparksee::gdb::type_t t)
Allows connectivity through nodes of the given type.
- virtual void [AddAllNodeTypes](#) ()
Allows connectivity through all node types of the graph.
- virtual void [ExcludeNodes](#) (sparksee::gdb::Objects &nodes)
Set which nodes can't be used.
- virtual void [ExcludeEdges](#) (sparksee::gdb::Objects &edges)
Set which edges can't be used.
- [ConnectedComponents](#) * [GetConnectedComponents](#) ()
Returns the results generated by the execution of the algorithm.
- virtual void [Run](#) ()=0
Runs the algorithm in order to find the connected components.
- void [SetMaterializedAttribute](#) (const std::wstring &attributeName)
Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Protected Types

- typedef std::map< [sparksee::gdb::type_t](#), [sparksee::gdb::EdgesDirection](#) > [EdgeTypes_t](#)
A type definition to store allowed edge types.
- typedef std::vector< [sparksee::gdb::type_t](#) > [NodeTypes_t](#)
A type definition to store allowed node types.

Protected Member Functions

- [StrongConnectivity](#) (sparksee::gdb::Session &s)
Creates a new instance of [StrongConnectivity](#).
- void [AssertAddedEdges](#) ()
Check that edges had been added.
- void [AssertAddedNodes](#) ()

Check that nodes had been added.

- void [AssertNotComputed](#) ()
Check that the connectivity had not been calculated yet.
- void [SetConnectedComponent](#) (sparksee::gdb::oid_t idNode)
Assigns the current component to the given node.
- void [SetNodesNotVisited](#) ()
Set all the selected nodes in nodesNotVisited.
- void [AssertNotComponentAttribute](#) (const std::wstring &attributeName)
Check that the given attribute name is not already in use.
- void [AssertComputed](#) ()
Check that the connectivity had been calculated.
- void [AssertEdgeType](#) (sparksee::gdb::type_t edgetype)
Check that the given edge type is valid.
- void [AssertNodeType](#) (sparksee::gdb::type_t nodetype)
Check that the given node type is valid.
- void [CreateGlobalPersistentAttribute](#) (const std::wstring &attributeName)
Set a new persistent global attribute to store the connectivity information.
- void [CreateGlobalTransientAttribute](#) ()
Set a new temporary global attribute to store the connectivity information.
- void [RemoveGlobalAttribute](#) ()
Removes the global attribute where the connectivity information is stored.
- sparksee::gdb::bool_t [IsNodeTypeAllowed](#) (sparksee::gdb::oid_t nodeId)
Check if the given node has an allowed type.
- sparksee::gdb::bool_t [IsNodeExcluded](#) (sparksee::gdb::oid_t node)
Check if the given node is forbidden.
- sparksee::gdb::bool_t [IsEdgeExcluded](#) (sparksee::gdb::oid_t edge)
Check if the given edge is forbidden.

Protected Attributes

- sparksee::gdb::Session * [sess](#)
Session.
- sparksee::gdb::Graph * [graph](#)
Graph.

- [EdgeTypes_t edgeTypes](#)
Allowed edge types.
- `std::vector< sparksee::gdb::type_t > nodeTypes`
Allowed node types.
- [sparksee::gdb::attr_t attrComponent](#)
common attribute where the connected component information is stored.
- `std::wstring attrComponentName`
name of the common attribute where the connected component information is stored.
- [sparksee::gdb::int64_t actualComponent](#)
Current component identifier.
- `sparksee::gdb::Objects * nodesNotVisited`
Identifiers of the nodes not visited.
- [sparksee::gdb::bool_t matResults](#)
Materialized results.
- [sparksee::gdb::bool_t computed](#)
True if the connectivity has been calculated.
- `sparksee::gdb::Objects * excludedNodes`
The set of excluded nodes.
- `sparksee::gdb::Objects * excludedEdges`
The set of excluded edges.
- [ConnectedComponents * ccs](#)
The calculated connectivity information.

5.56.1 Detailed Description

[StrongConnectivity](#) class.

Any class implementing this abstract class can be used to solve the problem of finding strongly connected components in a **directed** graph.

It consists in finding components where every pair (u,v) of nodes contained in it has a path from u to v using the specified direction for each edge type.

It is possible to set some restrictions after constructing a new instance of this class and before running it in order to limit the results.

After the execution, we can retrieve the results stored in an instance of the [ConnectedComponents](#) class using the `GetConnectedComponents` method.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.56.2 Constructor & Destructor Documentation

5.56.2.1 StrongConnectivity::StrongConnectivity (sparksee::gdb::Session & *s*) [protected]

Creates a new instance of [StrongConnectivity](#).

Parameters:

s [in] [Session](#) to get the graph from and calculate the connectivity

5.56.3 Member Function Documentation

5.56.3.1 virtual void StrongConnectivity::AddEdgeType (sparksee::gdb::type_t *t*, sparksee::gdb::EdgesDirection *d*) [virtual]

Allows connectivity through edges of the given type.

Parameters:

t [in] Edge type.

d [in] Edge direction.

Reimplemented from [Connectivity](#).

5.56.3.2 virtual void StrongConnectivity::AddAllEdgeTypes (sparksee::gdb::EdgesDirection *d*) [virtual]

Allows connectivity through all edge types of the graph.

Parameters:

d [in] Edge direction.

Reimplemented from [Connectivity](#).

5.56.3.3 virtual void Connectivity::ExcludeNodes (sparksee::gdb::Objects & *nodes*) [virtual, inherited]

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters:

nodes [in] A set of node identifiers that must be kept intact until the destruction of the class.

5.56.3.4 virtual void Connectivity::ExcludeEdges (sparksee::gdb::Objects & *edges*) [virtual, inherited]

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters:

edges [in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.56.3.5 ConnectedComponents* Connectivity::GetConnectedComponents () [inherited]

Returns the results generated by the execution of the algorithm.

These results contain information related to the connected components found as the number of different components, the set of nodes contained in each component or many other data.

Returns:

Returns an instance of the class [ConnectedComponents](#) which contain information related to the connected components found.

5.56.3.6 virtual void Connectivity::Run () [pure virtual, inherited]

Runs the algorithm in order to find the connected components.

This method can be called only once.

Implemented in [StrongConnectivityGabow](#), and [WeakConnectivityDFS](#).

5.56.3.7 void Connectivity::SetMaterializedAttribute (const std::wstring & attributeName) [inherited]

Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it is only necessary to create a new instance of the class [ConnectedComponents](#) indicating the graph and the name of the common attribute type which stores the results. This instance will have all the information related to the connected components found in the moment of the execution of the algorithm that stored this data.

It is possible to run the algorithm without specifying this parameter in order to avoid materializing the results of the execution.

Parameters:

attributeName [in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

5.56.3.8 void Connectivity::SetNodesNotVisited () [protected, inherited]

Set all the selected nodes in nodesNotVisited.

That's all the nodes of the allowed node types but not the excluded ones.

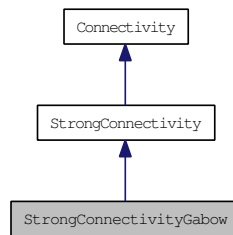
The documentation for this class was generated from the following file:

- StrongConnectivity.h

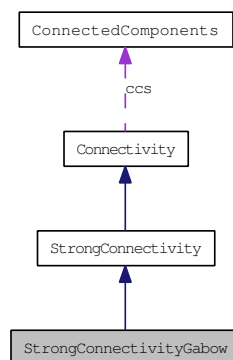
5.57 StrongConnectivityGabow Class Reference

This class can be used to solve the problem of finding strongly connected components in a **directed** graph.

Inheritance diagram for StrongConnectivityGabow:



Collaboration diagram for StrongConnectivityGabow:



Public Member Functions

- [StrongConnectivityGabow](#) (sparksee::gdb::Session &s)
Creates a new instance of [StrongConnectivityGabow](#).
- virtual [~StrongConnectivityGabow](#) ()
Destructor.
- void [Run](#) ()
Executes the algorithm.
- virtual void [AddEdgeType](#) (sparksee::gdb::type_t t, sparksee::gdb::EdgesDirection d)
Allows connectivity through edges of the given type.
- virtual void [AddAllEdgeTypes](#) (sparksee::gdb::EdgesDirection d)
Allows connectivity through all edge types of the graph.
- virtual void [AddNodeType](#) (sparksee::gdb::type_t t)
Allows connectivity through nodes of the given type.
- virtual void [AddAllNodeTypes](#) ()
Allows connectivity through all node types of the graph.
- virtual void [ExcludeNodes](#) (sparksee::gdb::Objects &nodes)

Set which nodes can't be used.

- virtual void [ExcludeEdges](#) (sparksee::gdb::Objects &edges)

Set which edges can't be used.

- [ConnectedComponents](#) * [GetConnectedComponents](#) ()

Returns the results generated by the execution of the algorithm.

- void [SetMaterializedAttribute](#) (const std::wstring &attributeName)

Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Protected Types

- typedef std::map< [sparksee::gdb::type_t](#), [sparksee::gdb::EdgesDirection](#) > [EdgeTypes_t](#)

A type definition to store allowed edge types.

- typedef std::vector< [sparksee::gdb::type_t](#) > [NodeTypes_t](#)

A type definition to store allowed node types.

Protected Member Functions

- void [AssertAddedEdges](#) ()

Check that edges had been added.

- void [AssertAddedNodes](#) ()

Check that nodes had been added.

- void [AssertNotComputed](#) ()

Check that the connectivity had not been calculated yet.

- void [SetConnectedComponent](#) (sparksee::gdb::oid_t idNode)

Assigns the current component to the given node.

- void [SetNodesNotVisited](#) ()

Set all the selected nodes in nodesNotVisited.

- void [AssertNotComponentAttribute](#) (const std::wstring &attributeName)

Check that the given attribute name is not already in use.

- void [AssertComputed](#) ()

Check that the connectivity had been calculated.

- void [AssertEdgeType](#) (sparksee::gdb::type_t edgetype)

Check that the given edge type is valid.

- void [AssertNodeType](#) (sparksee::gdb::type_t nodetype)

Check that the given node type is valid.

- void [CreateGlobalPersistentAttribute](#) (const std::wstring &attributeName)
Set a new persistent global attribute to store the connectivity information.
- void [CreateGlobalTransientAttribute](#) ()
Set a new temporary global attribute to store the connectivity information.
- void [RemoveGlobalAttribute](#) ()
Removes the global attribute where the connectivity information is stored.
- [sparksee::gdb::bool_t IsNodeTypeAllowed](#) ([sparksee::gdb::oid_t](#) nodeId)
Check if the given node has an allowed type.
- [sparksee::gdb::bool_t IsNodeExcluded](#) ([sparksee::gdb::oid_t](#) node)
Check if the given node is forbidden.
- [sparksee::gdb::bool_t IsEdgeExcluded](#) ([sparksee::gdb::oid_t](#) edge)
Check if the given edge is forbidden.

Protected Attributes

- [sparksee::gdb::Session](#) * [sess](#)
Session.
- [sparksee::gdb::Graph](#) * [graph](#)
Graph.
- [EdgeTypes_t](#) [edgeTypes](#)
Allowed edge types.
- std::vector< [sparksee::gdb::type_t](#) > [nodeTypes](#)
Allowed node types.
- [sparksee::gdb::attr_t](#) [attrComponent](#)
common attribute where the connected component information is stored.
- std::wstring [attrComponentName](#)
name of the common attribute where the connected component information is stored.
- [sparksee::gdb::int64_t](#) [actualComponent](#)
Current component identifier.
- [sparksee::gdb::Objects](#) * [nodesNotVisited](#)
Identifiers of the nodes not visited.
- [sparksee::gdb::bool_t](#) [matResults](#)
Materialized results.

- `sparksee::gdb::bool_t` `computed`
True if the connectivity has been calculated.
- `sparksee::gdb::Objects *` `excludedNodes`
The set of excluded nodes.
- `sparksee::gdb::Objects *` `excludedEdges`
The set of excluded edges.
- `ConnectedComponents *` `ccs`
The calculated connectivity information.

Classes

- class **InfoNode**

5.57.1 Detailed Description

This class can be used to solve the problem of finding strongly connected components in a **directed** graph.

It consists in finding components where every pair (u,v) of nodes contained in it has a path from u to v using the specified direction for each edge type. This implementation is based on the Gabow algorithm.

It is possible to set some restrictions after constructing a new instance of this class and before running it in order to limit the results.

After the execution, we can retrieve the results stored in an instance of the `ConnectedComponents` class using the `GetConnectedComponents` method.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.57.2 Constructor & Destructor Documentation

5.57.2.1 StrongConnectivityGabow::StrongConnectivityGabow (sparksee::gdb::Session & s)

Creates a new instance of `StrongConnectivityGabow`.

After creating this instance is required to indicate the set of edge types and the set of node types which will be navigated through while traversing the graph in order to find the strong connected components.

Parameters:

`s` [in] `Session` to get the graph from and calculate the connectivity

5.57.3 Member Function Documentation

5.57.3.1 virtual void StrongConnectivity::AddEdgeType (sparksee::gdb::type_t t, sparksee::gdb::EdgesDirection d) [virtual, inherited]

Allows connectivity through edges of the given type.

Parameters:

- t* [in] Edge type.
- d* [in] Edge direction.

Reimplemented from [Connectivity](#).

5.57.3.2 virtual void StrongConnectivity::AddAllEdgeTypes (sparksee::gdb::EdgesDirection *d*)
[virtual, inherited]

Allows connectivity through all edge types of the graph.

Parameters:

- d* [in] Edge direction.

Reimplemented from [Connectivity](#).

5.57.3.3 virtual void Connectivity::ExcludeNodes (sparksee::gdb::Objects & *nodes*) [virtual, inherited]

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters:

- nodes* [in] A set of node identifiers that must be kept intact until the destruction of the class.

5.57.3.4 virtual void Connectivity::ExcludeEdges (sparksee::gdb::Objects & *edges*) [virtual, inherited]

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters:

- edges* [in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.57.3.5 ConnectedComponents* Connectivity::GetConnectedComponents () [inherited]

Returns the results generated by the execution of the algorithm.

These results contain information related to the connected components found as the number of different components, the set of nodes contained in each component or many other data.

Returns:

- Returns an instance of the class [ConnectedComponents](#) which contain information related to the connected components found.

5.57.3.6 void Connectivity::SetMaterializedAttribute (const std::wstring & *attributeName*) [inherited]

Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it is only necessary to create a new instance of the class [ConnectedComponents](#) indicating the graph and the name of the common attribute type which stores the results. This instance will have all the information related to the connected components found in the moment of the execution of the algorithm that stored this data.

It is possible to run the algorithm without specifying this parameter in order to avoid materializing the results of the execution.

Parameters:

attributeName [in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

5.57.3.7 void Connectivity::SetNodesNotVisited () [protected, inherited]

Set all the selected nodes in nodesNotVisited.

That's all the nodes of the allowed node types but not the excluded ones.

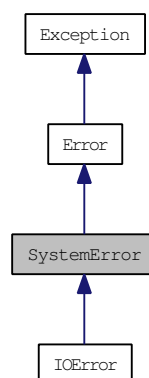
The documentation for this class was generated from the following file:

- StrongConnectivityGabow.h

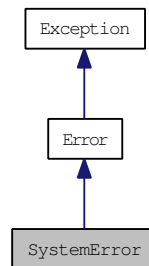
5.58 SystemError Class Reference

System error class.

Inheritance diagram for SystemError:



Collaboration diagram for SystemError:



Public Member Functions

- [SystemError](#) ()
Creates a new instance.
- [SystemError](#) (const std::string &mess)
Creates a new instance.
- virtual [~SystemError](#) ()
Destructor.
- const std::string & [Message](#) () const
Gets the message of the exception.
- void [SetMessage](#) (const std::string &mess)
Sets the message of the exception.

Static Public Member Functions

- static [Error NewError](#) (int32_t coreErrorCode)
Creates a new [Error](#) instance from a sparksee_core error code.

Protected Attributes

- std::string [message](#)
Message of the exception.

5.58.1 Detailed Description

System error class.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.58.2 Constructor & Destructor Documentation

5.58.2.1 `SystemError::SystemError (const std::string & mess)`

Creates a new instance.

Parameters:

mess [in] Message of the exception.

5.58.3 Member Function Documentation

5.58.3.1 `static Error Error::NewError (int32_t coreErrorCode)` [static, inherited]

Creates a new [Error](#) instance from a sparksee_core error code.

Parameters:

coreErrorCode [in] Sparkseecore error code.

Returns:

Depending on the given sparksee_core error, this may return an [Error](#) instance or an specific [Error](#) subclass instance.

5.58.3.2 `const std::string& Exception::Message () const` [inherited]

Gets the message of the exception.

It should be called GetMessage but this is not possible because of a macro defined in Windows.

Returns:

The message of the exception.

5.58.3.3 `void Exception::SetMessage (const std::string & mess)` [inherited]

Sets the message of the exception.

Parameters:

mess [in] Message.

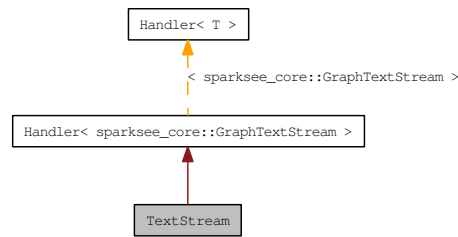
The documentation for this class was generated from the following file:

- Exception.h

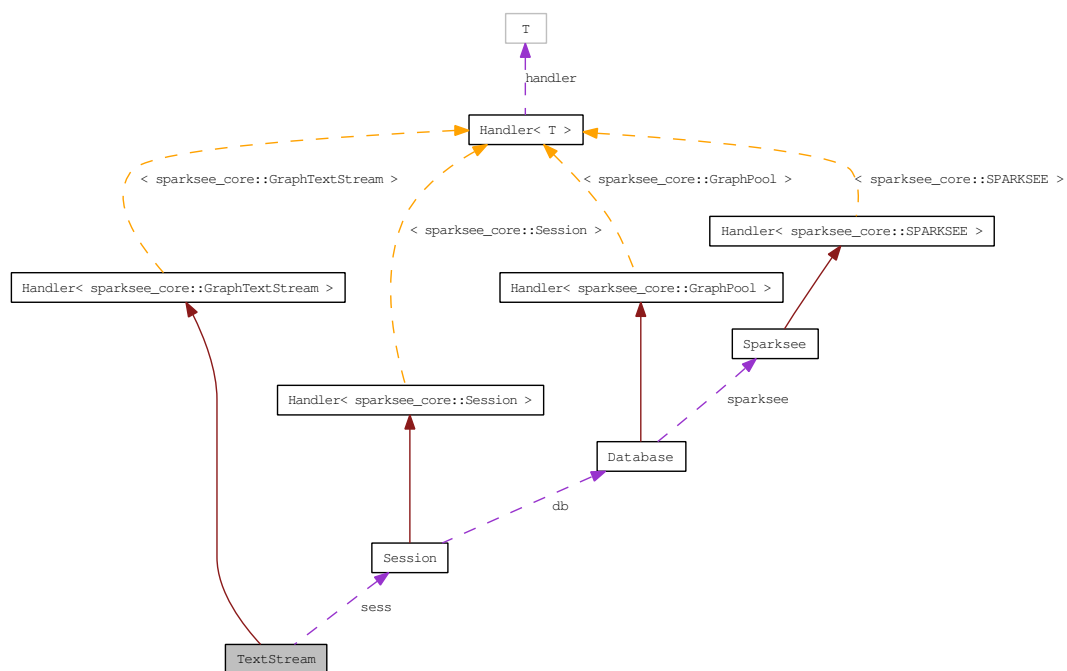
5.59 TextStream Class Reference

[TextStream](#) class.

Inheritance diagram for TextStream:



Collaboration diagram for TextStream:



Public Member Functions

- `TextStream` (`bool_t` append)
Creates a new instance.
- `int32_t Read` (`uchar_t` *dataOUT, `int32_t` length) const
Read data from the stream.
- `void Write` (const `uchar_t` *dataIN, `int32_t` length)
Write data to the stream.
- `void Close` ()
Closes the stream.
- `virtual ~TextStream` ()

Destructor.

- `bool_t IsNull () const`

Returns TRUE if the stream is not available.

Friends

- class `Graph`

5.59.1 Detailed Description

`TextStream` class.

It allows for reading and writing Text attribute values.

It is very important to close the stream once no more reading or writing operations will be performed to ensure data is successfully stored.

Whereas string attributes are set and got using the `Value` class, text attributes are operated using a stream pattern.

Use of `TextStream` for writing: (i) Create a `TextStream` instance and (ii) set the stream for a text attribute of a node or edge instance with the graph `SetAttributeText` method. Once the set attribute text has been done, (iii) perform as many write operations as you need to the `TextStream` instance. Lastly, (iv) execute `Close` to flush and close the stream.

Use of `TextStream` for reading: (i) Get the stream of a text attribute of a node or edge instance with the `GetAttributeText` graph method. Once you have the `TextStream` instance, (ii) you can execute `Read` operations to read from the stream. (iii) The end of the stream is reached when `Read` returns 0. Finally, (iv) execute `Close` to close stream resources.

Check out the 'Attributes and values' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.59.2 Constructor & Destructor Documentation

5.59.2.1 `TextStream::TextStream (bool_t append)`

Creates a new instance.

A `TextStream` only can be created by the user to write data.

Parameters:

append [in] If TRUE, the it is created in append mode to write from the end of the stream, otherwise it is created to write from the beginning of the stream.

5.59.3 Member Function Documentation

5.59.3.1 `int32_t TextStream::Read (uchar_t * dataOUT, int32_t length) const`

Read data from the stream.

Parameters:

dataOUT [out] Buffer to read data to. It must be allocated by the user.

length [in] Length of the given data buffer. It must be > 0 .

Returns:

Amount of read data (\leq length). If 0, there is no more data to be read from the stream.

5.59.3.2 void TextStream::Write (const uchar_t * dataIN, int32_t length)

Write data to the stream.

Parameters:

dataIN [in] Buffer to write data from.

length [in] Length of the data buffer. It must be > 0 .

5.59.3.3 void TextStream::Close ()

Closes the stream.

Once the Stream is closed, it cannot be used again.

5.59.3.4 bool_t TextStream::IsNull () const

Returns TRUE if the stream is not available.

It returns FALSE if the stream is ready for reading or writing data.

Returns:

FALSE if the stream is ready

Reimplemented from [Handler< sparksee_core::GraphTextStream >](#).

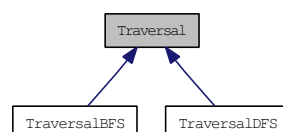
The documentation for this class was generated from the following file:

- Stream.h

5.60 Traversal Class Reference

[Traversal](#) class.

Inheritance diagram for Traversal:



Public Member Functions

- virtual void [AddEdgeType](#) (sparksee::gdb::type_t t, sparksee::gdb::EdgesDirection d)
Allows for traversing edges of the given type.
- virtual void [AddAllEdgeTypes](#) (sparksee::gdb::EdgesDirection d)
Allows for traversing all edge types of the graph.
- virtual void [AddNodeType](#) (sparksee::gdb::type_t t)
Allows for traversing nodes of the given type.
- virtual void [AddAllNodeTypes](#) ()
Allows for traversing all node types of the graph.
- virtual void [ExcludeNodes](#) (sparksee::gdb::Objects &nodes)
Set which nodes can't be used.
- virtual void [ExcludeEdges](#) (sparksee::gdb::Objects &edges)
Set which edges can't be used.
- virtual sparksee::gdb::oid_t [Next](#) ()=0
Gets the next object of the traversal.
- virtual sparksee::gdb::bool_t [HasNext](#) ()=0
Gets if there are more objects to be traversed.
- virtual sparksee::gdb::int32_t [GetCurrentDepth](#) () const =0
Returns the depth of the current node.
- virtual void [SetMaximumHops](#) (sparksee::gdb::int32_t maxhops)
Sets the maximum hops restriction.
- virtual [~Traversal](#) ()
Destructor.

Protected Member Functions

- [Traversal](#) (sparksee::gdb::Session &s, sparksee::gdb::oid_t node)
Creates a new instance.
- void [AssertAddedEdges](#) ()
Check that edges had been added.
- void [AssertAddedNodes](#) ()
Check that nodes had been added.
- void [AssertEdgeType](#) (sparksee::gdb::type_t edgetype)
Check that the given edge type is valid.

- void [AssertNodeType](#) (sparksee::gdb::type_t nodetype)
Check that the given node type is valid.
- sparksee::gdb::bool_t [IsNodeTypeAllowed](#) (sparksee::gdb::oid_t nodeId)
Check if the given node has an allowed type.
- sparksee::gdb::bool_t [IsNodeExcluded](#) (sparksee::gdb::oid_t node)
Check if the given node is forbidden.
- sparksee::gdb::bool_t [IsEdgeExcluded](#) (sparksee::gdb::oid_t edge)
Check if the given edge is forbidden.

Protected Attributes

- sparksee::gdb::Session * [sess](#)
Session.
- sparksee::gdb::Graph * [graph](#)
Graph.
- sparksee::gdb::oid_t [src](#)
Source node of the traversal.
- std::map< [sparksee::gdb::type_t](#), [sparksee::gdb::EdgesDirection](#) > [edgeTypes](#)
Allowed edge types.
- std::vector< [sparksee::gdb::type_t](#) > [nodeTypes](#)
Allowed node types.
- sparksee::gdb::int32_t [maxHops](#)
Maximum number of hops allowed.
- sparksee::gdb::Objects * [excludedNodes](#)
The set of excluded nodes.
- sparksee::gdb::Objects * [excludedEdges](#)
The set of excluded edges.

5.60.1 Detailed Description

[Traversal](#) class.

Any class implementing this abstract class can be used to traverse a graph.

Once the instance has been created and the allowed node and edge types has been set, it can be used as an iterator, retrieving the next object identifier of the traversal until there are no more.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.60.2 Constructor & Destructor Documentation

5.60.2.1 `Traversal::Traversal (sparksee::gdb::Session & s, sparksee::gdb::oid_t node)` `[protected]`

Creates a new instance.

Parameters:

s [in] [Session](#) to get the graph from and perform traversal.
node [in] Node to start traversal from.

5.60.3 Member Function Documentation

5.60.3.1 `virtual void Traversal::AddEdgeType (sparksee::gdb::type_t t, sparksee::gdb::EdgesDirection d)` `[virtual]`

Allows for traversing edges of the given type.

Parameters:

t [in] Edge type.
d [in] Edge direction.

5.60.3.2 `virtual void Traversal::AddAllEdgeTypes (sparksee::gdb::EdgesDirection d)` `[virtual]`

Allows for traversing all edge types of the graph.

Parameters:

d [in] Edge direction.

5.60.3.3 `virtual void Traversal::ExcludeNodes (sparksee::gdb::Objects & nodes)` `[virtual]`

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters:

nodes [in] A set of node identifiers that must be kept intact until the destruction of the class.

5.60.3.4 `virtual void Traversal::ExcludeEdges (sparksee::gdb::Objects & edges)` `[virtual]`

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters:

edges [in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.60.3.5 `virtual sparksee::gdb::oid_t Traversal::Next () [pure virtual]`

Gets the next object of the traversal.

Returns:

A node or edge identifier.

Implemented in [TraversalBFS](#), and [TraversalDFS](#).

5.60.3.6 `virtual sparksee::gdb::bool_t Traversal::HasNext () [pure virtual]`

Gets if there are more objects to be traversed.

Returns:

TRUE if there are more objects, FALSE otherwise.

Implemented in [TraversalBFS](#), and [TraversalDFS](#).

5.60.3.7 `virtual sparksee::gdb::int32_t Traversal::GetCurrentDepth () const [pure virtual]`

Returns the depth of the current node.

That is, it returns the depth of the node returned in the last call to [Next\(\)](#).

Returns:

The depth of the current node.

Implemented in [TraversalBFS](#), and [TraversalDFS](#).

5.60.3.8 `virtual void Traversal::SetMaximumHops (sparksee::gdb::int32_t maxhops) [virtual]`

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

Parameters:

maxhops [in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

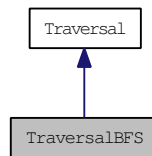
The documentation for this class was generated from the following file:

- [Traversal.h](#)

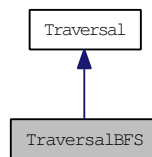
5.61 TraversalBFS Class Reference

Breadth-First Search implementation of [Traversal](#).

Inheritance diagram for TraversalBFS:



Collaboration diagram for TraversalBFS:



Public Member Functions

- virtual [sparksee::gdb::oid_t Next \(\)](#)
Gets the next object of the traversal.
- virtual [sparksee::gdb::bool_t HasNext \(\)](#)
Gets if there are more objects to be traversed.
- virtual [~TraversalBFS \(\)](#)
Destructor.
- virtual [sparksee::gdb::int32_t GetCurrentDepth \(\) const](#)
Returns the depth of the current node.
- [TraversalBFS](#) ([sparksee::gdb::Session &s](#), [sparksee::gdb::oid_t node](#))
Creates a new instance.
- virtual void [AddEdgeType](#) ([sparksee::gdb::type_t t](#), [sparksee::gdb::EdgesDirection d](#))
Allows for traversing edges of the given type.
- virtual void [AddAllEdgeTypes](#) ([sparksee::gdb::EdgesDirection d](#))
Allows for traversing all edge types of the graph.
- virtual void [AddNodeType](#) ([sparksee::gdb::type_t t](#))
Allows for traversing nodes of the given type.
- virtual void [AddAllNodeTypes](#) ()
Allows for traversing all node types of the graph.
- virtual void [ExcludeNodes](#) ([sparksee::gdb::Objects &nodes](#))
Set which nodes can't be used.

- virtual void [ExcludeEdges](#) (sparksee::gdb::Objects &edges)
Set which edges can't be used.
- virtual void [SetMaximumHops](#) (sparksee::gdb::int32_t maxhops)
Sets the maximum hops restriction.

Protected Member Functions

- void [AssertAddedEdges](#) ()
Check that edges had been added.
- void [AssertAddedNodes](#) ()
Check that nodes had been added.
- void [AssertEdgeType](#) (sparksee::gdb::type_t edgetype)
Check that the given edge type is valid.
- void [AssertNodeType](#) (sparksee::gdb::type_t nodetype)
Check that the given node type is valid.
- sparksee::gdb::bool_t [IsNodeTypeAllowed](#) (sparksee::gdb::oid_t nodeId)
Check if the given node has an allowed type.
- sparksee::gdb::bool_t [IsNodeExcluded](#) (sparksee::gdb::oid_t node)
Check if the given node is forbidden.
- sparksee::gdb::bool_t [IsEdgeExcluded](#) (sparksee::gdb::oid_t edge)
Check if the given edge is forbidden.

Protected Attributes

- sparksee::gdb::Session * [sess](#)
Session.
- sparksee::gdb::Graph * [graph](#)
Graph.
- sparksee::gdb::oid_t [src](#)
Source node of the traversal.
- std::map< sparksee::gdb::type_t, sparksee::gdb::EdgesDirection > [edgeTypes](#)
Allowed edge types.
- std::vector< sparksee::gdb::type_t > [nodeTypes](#)
Allowed node types.
- sparksee::gdb::int32_t [maxHops](#)

Maximum number of hops allowed.

- sparksee::gdb::Objects * [excludedNodes](#)

The set of excluded nodes.

- sparksee::gdb::Objects * [excludedEdges](#)

The set of excluded edges.

5.61.1 Detailed Description

Breadth-First Search implementation of [Traversal](#).

Starting from a source node, it visits all its neighbors at distance 1, then all its neighbors at distance 2, and so on.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.61.2 Constructor & Destructor Documentation

5.61.2.1 TraversalBFS::TraversalBFS (sparksee::gdb::Session & s, sparksee::gdb::oid_t node)

Creates a new instance.

Parameters:

s [in] [Session](#) to get the graph from and perform traversal.

node [in] Node to start traversal from.

5.61.3 Member Function Documentation

5.61.3.1 virtual sparksee::gdb::oid_t TraversalBFS::Next () [virtual]

Gets the next object of the traversal.

Returns:

A node or edge identifier.

Implements [Traversal](#).

5.61.3.2 virtual sparksee::gdb::bool_t TraversalBFS::HasNext () [virtual]

Gets if there are more objects to be traversed.

Returns:

TRUE if there are more objects, FALSE otherwise.

Implements [Traversal](#).

5.61.3.3 virtual sparksee::gdb::int32_t TraversalBFS::GetCurrentDepth () const [virtual]

Returns the depth of the current node.

That is, it returns the depth of the node returned in the last call to [Next\(\)](#).

Returns:

The depth of the current node.

Implements [Traversal](#).

5.61.3.4 virtual void Traversal::AddEdgeType (sparksee::gdb::type_t *t*, sparksee::gdb::EdgesDirection *d*) [virtual, inherited]

Allows for traversing edges of the given type.

Parameters:

t [in] Edge type.

d [in] Edge direction.

5.61.3.5 virtual void Traversal::AddAllEdgeTypes (sparksee::gdb::EdgesDirection *d*) [virtual, inherited]

Allows for traversing all edge types of the graph.

Parameters:

d [in] Edge direction.

5.61.3.6 virtual void Traversal::ExcludeNodes (sparksee::gdb::Objects & *nodes*) [virtual, inherited]

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters:

nodes [in] A set of node identifiers that must be kept intact until the destruction of the class.

5.61.3.7 virtual void Traversal::ExcludeEdges (sparksee::gdb::Objects & *edges*) [virtual, inherited]

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters:

edges [in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.61.3.8 `virtual void Traversal::SetMaximumHops (sparksee::gdb::int32_t maxhops)`
`[virtual, inherited]`

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

Parameters:

maxhops [in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

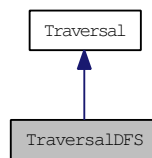
The documentation for this class was generated from the following file:

- TraversalBFS.h

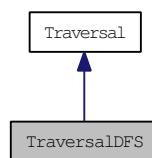
5.62 TraversalDFS Class Reference

Depth-First Search (DFS) implementation of [Traversal](#).

Inheritance diagram for TraversalDFS:



Collaboration diagram for TraversalDFS:



Public Member Functions

- virtual `sparksee::gdb::oid_t Next ()`
Gets the next object of the traversal.
- virtual `sparksee::gdb::bool_t HasNext ()`
Gets if there are more objects to be traversed.
- virtual `sparksee::gdb::int32_t GetCurrentDepth () const`
Returns the depth of the current node.
- virtual `~TraversalDFS ()`
Destructor.

- **TraversalDFS** (sparksee::gdb::Session &s, sparksee::gdb::oid_t node)
Creates a new instance.
- virtual void **AddEdgeType** (sparksee::gdb::type_t t, sparksee::gdb::EdgesDirection d)
Allows for traversing edges of the given type.
- virtual void **AddAllEdgeTypes** (sparksee::gdb::EdgesDirection d)
Allows for traversing all edge types of the graph.
- virtual void **AddNodeType** (sparksee::gdb::type_t t)
Allows for traversing nodes of the given type.
- virtual void **AddAllNodeTypes** ()
Allows for traversing all node types of the graph.
- virtual void **ExcludeNodes** (sparksee::gdb::Objects &nodes)
Set which nodes can't be used.
- virtual void **ExcludeEdges** (sparksee::gdb::Objects &edges)
Set which edges can't be used.
- virtual void **SetMaximumHops** (sparksee::gdb::int32_t maxhops)
Sets the maximum hops restriction.

Protected Member Functions

- void **AssertAddedEdges** ()
Check that edges had been added.
- void **AssertAddedNodes** ()
Check that nodes had been added.
- void **AssertEdgeType** (sparksee::gdb::type_t edgetype)
Check that the given edge type is valid.
- void **AssertNodeType** (sparksee::gdb::type_t nodetype)
Check that the given node type is valid.
- sparksee::gdb::bool_t **IsNodeTypeAllowed** (sparksee::gdb::oid_t nodeId)
Check if the given node has an allowed type.
- sparksee::gdb::bool_t **IsNodeExcluded** (sparksee::gdb::oid_t node)
Check if the given node is forbidden.
- sparksee::gdb::bool_t **IsEdgeExcluded** (sparksee::gdb::oid_t edge)
Check if the given edge is forbidden.

Protected Attributes

- sparksee::gdb::Session * [sess](#)
Session.
- sparksee::gdb::Graph * [graph](#)
Graph.
- [sparksee::gdb::oid_t src](#)
Source node of the traversal.
- std::map< [sparksee::gdb::type_t](#), [sparksee::gdb::EdgesDirection](#) > [edgeTypes](#)
Allowed edge types.
- std::vector< [sparksee::gdb::type_t](#) > [nodeTypes](#)
Allowed node types.
- [sparksee::gdb::int32_t maxHops](#)
Maximum number of hops allowed.
- sparksee::gdb::Objects * [excludedNodes](#)
The set of excluded nodes.
- sparksee::gdb::Objects * [excludedEdges](#)
The set of excluded edges.

Classes

- class **NeighborsInfo**
Store neighbors information.

5.62.1 Detailed Description

Depth-First Search (DFS) implementation of [Traversal](#).

Starting from a source or root node, it visits as far as possible along each branch before backtracking.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.62.2 Constructor & Destructor Documentation

5.62.2.1 TraversalDFS::TraversalDFS (sparksee::gdb::Session & s, sparksee::gdb::oid_t node)

Creates a new instance.

Parameters:

- s* [in] [Session](#) to get the graph from and perform traversal.
node [in] Node to start traversal from.

5.62.3 Member Function Documentation**5.62.3.1 virtual sparksee::gdb::oid_t TraversalDFS::Next () [virtual]**

Gets the next object of the traversal.

Returns:

A node or edge identifier.

Implements [Traversal](#).

5.62.3.2 virtual sparksee::gdb::bool_t TraversalDFS::HasNext () [virtual]

Gets if there are more objects to be traversed.

Returns:

TRUE if there are more objects, FALSE otherwise.

Implements [Traversal](#).

5.62.3.3 virtual sparksee::gdb::int32_t TraversalDFS::GetCurrentDepth () const [virtual]

Returns the depth of the current node.

That is, it returns the depth of the node returned in the last call to [Next\(\)](#).

Returns:

The depth of the current node.

Implements [Traversal](#).

5.62.3.4 virtual void Traversal::AddEdgeType (sparksee::gdb::type_t *t*, sparksee::gdb::EdgesDirection *d*) [virtual, inherited]

Allows for traversing edges of the given type.

Parameters:

- t* [in] Edge type.
d [in] Edge direction.

5.62.3.5 virtual void Traversal::AddAllEdgeTypes (sparksee::gdb::EdgesDirection *d*) [virtual, inherited]

Allows for traversing all edge types of the graph.

Parameters:

- d* [in] Edge direction.

5.62.3.6 virtual void Traversal::ExcludeNodes (sparksee::gdb::Objects & *nodes*) [virtual, inherited]

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters:

nodes [in] A set of node identifiers that must be kept intact until the destruction of the class.

5.62.3.7 virtual void Traversal::ExcludeEdges (sparksee::gdb::Objects & *edges*) [virtual, inherited]

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters:

edges [in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.62.3.8 virtual void Traversal::SetMaximumHops (sparksee::gdb::int32_t *maxhops*) [virtual, inherited]

Sets the maximum hops restriction.

All paths longer than the maximum hops restriction will be ignored.

Parameters:

maxhops [in] The maximum hops restriction. It must be positive or zero. Zero, the default value, means unlimited.

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

- TraversalDFS.h

5.63 Type Class Reference

[Type](#) data class.

Public Member Functions

- [~Type](#) ()
Destructor.
- [type_t GetId](#) () const
Gets the [Sparksee](#) type identifier.
- [ObjectType GetObjectType](#) () const
Gets the object type.

- `const std::wstring & GetName () const`
Gets the unique type name.
- `int64_t GetNumObjects () const`
Gets the number of objects belonging to the type.
- `bool_t GetIsDirected () const`
Gets if this is a directed edge type.
- `bool_t GetIsRestricted () const`
Gets if this is a restricted edge type.
- `bool_t GetAreNeighborsIndexed () const`
Gets if this is an edge type with neighbors index.
- `type_t GetRestrictedFrom () const`
Gets the tail or source type identifier for restricted edge types.
- `type_t GetRestrictedTo () const`
Gets the head or target type identifier for restricted edge types.

Static Public Attributes

- static const `type_t InvalidType`
Invalid type identifier constant.
- static const `type_t GlobalType`
Global type identifier constant.

Friends

- class `Graph`

5.63.1 Detailed Description

`Type` data class.

It contains information about a node or edge type.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.63.2 Member Function Documentation

5.63.2.1 `type_t Type::GetId () const` [inline]

Gets the [Sparksee](#) type identifier.

Returns:

The [Sparksee](#) type identifier.

5.63.2.2 `ObjectType Type::GetObjectType () const` [inline]

Gets the object type.

Returns:

The object type.

5.63.2.3 `const std::wstring& Type::GetName () const` [inline]

Gets the unique type name.

Returns:

The unique type name.

5.63.2.4 `int64_t Type::GetNumObjects () const` [inline]

Gets the number of objects belonging to the type.

Returns:

The number of objects belonging to the type.

5.63.2.5 `bool_t Type::GetIsDirected () const` [inline]

Gets if this is a directed edge type.

Returns:

TRUE for directed edge types, FALSE otherwise.

5.63.2.6 `bool_t Type::GetIsRestricted () const` [inline]

Gets if this is a restricted edge type.

Returns:

TRUE for restricted edge types, FALSE otherwise.

5.63.2.7 bool_t Type::GetAreNeighborsIndexed () const [inline]

Gets if this is an edge type with neighbors index.

Returns:

TRUE for edges types with neighbors index, FALSE otherwise.

5.63.2.8 type_t Type::GetRestrictedFrom () const [inline]

Gets the tail or source type identifier for restricted edge types.

Returns:

For restricted edge types, the tail or source type identifier, the [Type](#) InvalidType otherwise.

5.63.2.9 type_t Type::GetRestrictedTo () const [inline]

Gets the head or target type identifier for restricted edge types.

Returns:

For restricted edge types, the head or target type identifier, the [Type](#) InvalidType otherwise.

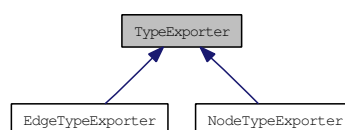
The documentation for this class was generated from the following file:

- Graph_data.h

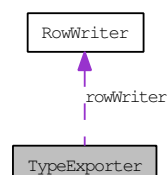
5.64 TypeExporter Class Reference

Base [TypeExporter](#) class.

Inheritance diagram for TypeExporter:



Collaboration diagram for TypeExporter:



Public Member Functions

- virtual [~TypeExporter](#) ()
Destructor.
- void [Register](#) ([TypeExporterListener](#) &tel)
Registers a new listener.
- virtual void [Run](#) ()=0 throw (sparksee::gdb::IOException, sparksee::gdb::Error)
Runs export process.
- void [SetRowWriter](#) ([RowWriter](#) &rw)
Sets the output data destination.
- void [SetGraph](#) (sparksee::gdb::Graph &g)
Sets the graph that will be exported.
- void [SetType](#) (sparksee::gdb::type_t t)
Sets the type to be exported.
- void [SetAttributes](#) (sparksee::gdb::AttributeList &attrs)
Sets the list of Attributes.
- void [SetFrequency](#) (sparksee::gdb::int32_t freq)
Sets the frequency of listener notification.
- void [SetHeader](#) (sparksee::gdb::bool_t header)
Sets the presence of a header row.

Protected Member Functions

- [TypeExporter](#) ()
Creates a new instance.
- [TypeExporter](#) ([RowWriter](#) &rw, sparksee::gdb::Graph &g, [sparksee::gdb::type_t](#) t, sparksee::gdb::AttributeList &attrs)
Creates a new instance with the minimum common required arguments.
- [sparksee::gdb::bool_t CanRun](#) ()
Checks that all the required settings are ready to run.
- void [NotifyListeners](#) (const [TypeExporterEvent](#) &ev)
Notifies progress to all registered listeners.
- void [RunProcess](#) () throw (sparksee::gdb::IOException, sparksee::gdb::Error)
Runs export process.
- void [SetHeadAttribute](#) (sparksee::gdb::attr_t attr)
Sets the attribute that will be used to get the value to be dumped for the head of the edge.

- void [SetHeadPosition](#) ([sparksee::gdb::int32_t](#) pos)
Sets the position (index column) of the head attribute in the exported data.
- void [SetTailAttribute](#) ([sparksee::gdb::attr_t](#) attr)
Sets the attribute that will be used to get the value to be dumped for the tail of the edge.
- void [SetTailPosition](#) ([sparksee::gdb::int32_t](#) pos)
Sets the position (index column) of the tail attribute in the exported data.

5.64.1 Detailed Description

Base [TypeExporter](#) class.

Base class to export a node or edge type from a graph using a [RowWriter](#).

[TypeExporterListener](#) can be registered to receive information about the progress of the export process by means of [TypeExporterEvent](#). The default frequency of notification to listeners is 100000.

By default no header row is created.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.64.2 Constructor & Destructor Documentation

5.64.2.1 [TypeExporter::TypeExporter](#) ([RowWriter](#) & *rw*, [sparksee::gdb::Graph](#) & *g*, [sparksee::gdb::type_t](#) *t*, [sparksee::gdb::AttributeList](#) & *attrs*) [[protected](#)]

Creates a new instance with the minimum common required arguments.

Parameters:

- rw* [in] Output [RowWriter](#).
- g* [in] [Graph](#).
- t* [in] [Type](#) identifier.
- attrs* [in] [Attribute](#) identifiers to be exported.

5.64.3 Member Function Documentation

5.64.3.1 [sparksee::gdb::bool_t](#) [TypeExporter::CanRun](#) () [[protected](#)]

Checks that all the required settings are ready to run.

Returns:

Returns true if all the settings are ready.

5.64.3.2 void TypeExporter::NotifyListeners (const TypeExporterEvent & *ev*) [protected]

Notifies progress to all registered listeners.

Parameters:

ev [in] Progress event to be notified.

5.64.3.3 void TypeExporter::RunProcess () throw (sparksee::gdb::IOException, sparksee::gdb::Error) [protected]

Runs export process.

Exceptions:

[IOException](#) If bad things happen writting to the [RowWriter](#).

5.64.3.4 void TypeExporter::SetHeadAttribute (sparksee::gdb::attr_t *attr*) [protected]

Sets the attribute that will be used to get the value to be dumped for the head of the edge.

This method is protected because only the Edge exporters should have it.

Parameters:

attr [in] Head [Attribute](#)

Reimplemented in [EdgeTypeExporter](#).

5.64.3.5 void TypeExporter::SetHeadPosition (sparksee::gdb::int32_t *pos*) [protected]

Sets the position (index column) of the head attribute in the exported data.

This method is protected because only the Edge exporters should have it.

Parameters:

pos [in] Head position

Reimplemented in [EdgeTypeExporter](#).

5.64.3.6 void TypeExporter::SetTailAttribute (sparksee::gdb::attr_t *attr*) [protected]

Sets the attribute that will be used to get the value to be dumped for the tail of the edge.

This method is protected because only the Edge exporters should have it.

Parameters:

attr [in] Tail [Attribute](#)

Reimplemented in [EdgeTypeExporter](#).

5.64.3.7 void TypeExporter::SetTailPosition (sparksee::gdb::int32_t *pos*) [protected]

Sets the position (index column) of the tail attribute in the exported data.

This method is protected because only the Edge exporters should have it.

Parameters:

pos [in] Tail position

Reimplemented in [EdgeTypeExporter](#).

5.64.3.8 void TypeExporter::Register (TypeExporterListener & *tel*)

Registers a new listener.

Parameters:

tel [in] [TypeExporterListener](#) to be registered.

5.64.3.9 virtual void TypeExporter::Run () throw (sparksee::gdb::IOException, sparksee::gdb::Error) [pure virtual]

Runs export process.

Exceptions:

[IOException](#) If bad things happen writing to the [RowWriter](#).

Implemented in [EdgeTypeExporter](#), and [NodeTypeExporter](#).

5.64.3.10 void TypeExporter::SetRowWriter (RowWriter & *rw*)

Sets the output data destination.

Parameters:

rr [in] Input [RowWriter](#).

5.64.3.11 void TypeExporter::SetGraph (sparksee::gdb::Graph & *g*)

Sets the graph that will be exported.

Parameters:

g [in] [Graph](#).

5.64.3.12 void TypeExporter::SetType (sparksee::gdb::type_t *t*)

Sets the type to be exported.

Parameters:

t [in] [Type](#) identifier.

5.64.3.13 void TypeExporter::SetAttributes (sparksee::gdb::AttributeList & *attrs*)

Sets the list of Attributes.

Parameters:

attrs [in] [Attribute](#) identifiers to be exported

5.64.3.14 void TypeExporter::SetFrequency (sparksee::gdb::int32_t *freq*)

Sets the frequency of listener notification.

freq [in] Frequency in number of rows managed to notify progress to all listeners

5.64.3.15 void TypeExporter::SetHeader (sparksee::gdb::bool_t *header*)

Sets the presence of a header row.

Parameters:

header [in] If TRUE, a header row is dumped with the name of the attributes.

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

- [TypeExporter.h](#)

5.65 TypeExporterEvent Class Reference

Provides information about the progress of an TypeExproter instance.

Public Member Functions

- virtual [~TypeExporterEvent](#) ()
Destructor.
- [sparksee::gdb::type_t GetTypeId](#) () const
Gets the type identifier.
- [sparksee::gdb::int64_t GetCount](#) () const
Gets the current number of objects exported.
- [sparksee::gdb::int64_t GetTotal](#) () const
Gets the total number of objects exported.
- [sparksee::gdb::bool_t IsLast](#) () const
Gets if this is the last event or not.

Friends

- class [TypeExporter](#)

5.65.1 Detailed Description

Provides information about the progress of an TypeExproter instance.

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.65.2 Member Function Documentation

5.65.2.1 `sparksee::gdb::type_t TypeExporterEvent::GetTypeId () const` [inline]

Gets the type identifier.

Returns:

The type identifier.

5.65.2.2 `sparksee::gdb::int64_t TypeExporterEvent::GetCount () const` [inline]

Gets the current number of objects exported.

Returns:

The current number of objects exported.

5.65.2.3 `sparksee::gdb::int64_t TypeExporterEvent::GetTotal () const` [inline]

Gets the total number of objects exported.

Returns:

The total number of objects exported.

5.65.2.4 `sparksee::gdb::bool_t TypeExporterEvent::IsLast () const` [inline]

Gets if this is the last event or not.

Returns:

TRUE if this is the last event, FALSE otherwise.

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

- TypeExporter.h

5.66 TypeExporterListener Class Reference

Interface to be implemented to receive [TypeExporterEvent](#) events from a [TypeExporter](#).

Public Member Functions

- virtual void [NotifyEvent](#) (const [TypeExporterEvent](#) &tee)=0
Method to be notified from a [TypeExporter](#).
- virtual [~TypeExporterListener](#) ()
Destructor.

Protected Member Functions

- [TypeExporterListener](#) ()
Protected because none should instantiate a [RowWriter](#).

5.66.1 Detailed Description

Interface to be implemented to receive [TypeExporterEvent](#) events from a [TypeExporter](#).

Check out the 'Data export' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.66.2 Constructor & Destructor Documentation**5.66.2.1 TypeExporterListener::TypeExporterListener () [inline, protected]**

Protected because none should instantiate a [RowWriter](#).

Just inherited classes may use this empty constructor.

5.66.3 Member Function Documentation**5.66.3.1 virtual void TypeExporterListener::NotifyEvent (const TypeExporterEvent & tee) [pure virtual]**

Method to be notified from a [TypeExporter](#).

Parameters:

tee [in] Notified event.

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

- [TypeExporter.h](#)

5.67 TypeList Class Reference

[Sparksee](#) type identifier list.

Public Member Functions

- `int32_t Count () const`
Number of elements in the list.
- `TypeListIterator * Iterator ()`
Gets a new [TypeListIterator](#).
- `TypeList ()`
Constructor.
- `TypeList (const std::vector< type_t > &v)`
Constructor.
- `void Add (type_t type)`
Adds a [Sparksee](#) type identifier at the end of the list.
- `void Clear ()`
Clears the list.
- `~TypeList ()`
Destructor.

5.67.1 Detailed Description

[Sparksee](#) type identifier list.

It stores a [Sparksee](#) node or edge type identifier list.

Use [TypeListIterator](#) to access all elements into this collection.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.67.2 Constructor & Destructor Documentation

5.67.2.1 TypeList::TypeList ()

Constructor.

This creates an empty list.

5.67.2.2 TypeList::TypeList (const std::vector< [type_t](#) > &v)

Constructor.

Parameters:

v [in] Vector.

5.67.3 Member Function Documentation

5.67.3.1 `int32_t TypeList::Count () const` [inline]

Number of elements in the list.

Returns:

Number of elements in the list.

5.67.3.2 `TypeListIterator* TypeList::Iterator ()`

Gets a new [TypeListIterator](#).

Returns:

[TypeListIterator](#) instance.

5.67.3.3 `void TypeList::Add (type_t type)` [inline]

Adds a [Sparksee](#) type identifier at the end of the list.

Parameters:

type [in] [Sparksee](#) type identifier.

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

- `Graph_data.h`

5.68 `TypeListIterator` Class Reference

[TypeList](#) iterator class.

Public Member Functions

- [~TypeListIterator \(\)](#)
Destructor.
- [type_t Next \(\)](#)
Moves to the next element.
- [bool_t HasNext \(\)](#)
Gets if there are more elements.

Friends

- class [TypeList](#)

5.68.1 Detailed Description

TypeList iterator class.

Iterator to traverse all the Sparksee node or edge type identifiers into a TypeList instance.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.68.2 Member Function Documentation

5.68.2.1 type_t TypeListIterator::Next () [inline]

Moves to the next element.

Returns:

The next element.

5.68.2.2 bool_t TypeListIterator::HasNext () [inline]

Gets if there are more elements.

Returns:

TRUE if there are more elements, FALSE otherwise.

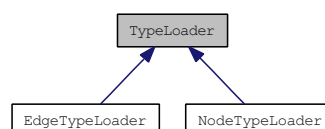
The documentation for this class was generated from the following file:

- Graph_data.h

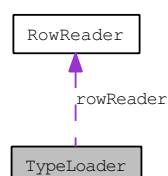
5.69 TypeLoader Class Reference

Base TypeLoader class.

Inheritance diagram for TypeLoader:



Collaboration diagram for TypeLoader:



Public Member Functions

- void [SetLogError](#) (const std::wstring &path) throw (sparksee::gdb::IOException)
Sets a log error file.
- void [SetLogOff](#) ()
Truns off all the error reporting.
- virtual [~TypeLoader](#) ()
Destructor.
- void [Register](#) (TypeLoaderListener &tel)
Registers a new listener.
- virtual void [Run](#) ()=0 throw (sparksee::gdb::IOException, sparksee::gdb::Error)
Run the loader.
- virtual void [RunTwoPhases](#) ()=0 throw (sparksee::gdb::IOException, sparksee::gdb::Error)
Run the loader for two phases loading.
- virtual void [RunNPhases](#) (sparksee::gdb::int32_t partitions)=0 throw (sparksee::gdb::IOException, sparksee::gdb::Error)
Run the loader for N phases loading.
- void [SetRowReader](#) (RowReader &rr)
Sets the input data source.
- void [SetGraph](#) (sparksee::gdb::Graph &g)
Sets the graph where the data will be loaded.
- void [SetLocale](#) (const std::wstring &localeStr)
Sets the locale that will be used to read the data.
- void [SetType](#) (sparksee::gdb::type_t t)
Sets the type to be loaded.
- void [SetAttributes](#) (sparksee::gdb::AttributeList &attrs)
Sets the list of Attributes.
- void [SetAttributePositions](#) (sparksee::gdb::Int32List &attrsPos)
Sets the list of attribute positions.
- void [SetTimestampFormat](#) (const std::wstring ×tampFormat)
Sets a specific timestamp format.
- void [SetFrequency](#) (sparksee::gdb::int32_t freq)
Sets the frequency of listener notification.

Protected Types

- enum [Mode](#) {
[ONE_PHASE](#),
[TWO_PHASES](#),
[N_PHASES](#) }

Load can work in different ways.

Protected Member Functions

- [sparksee::gdb::bool_t CanRun \(\)](#)
Checks that all the required settings are ready to run.
- void [Run \(Mode ph, sparksee::gdb::int32_t par\) throw \(sparksee::gdb::IOException, sparksee::gdb::Error\)](#)
Runs load process.
- [TypeLoader \(RowReader &rr, sparksee::gdb::Graph &g, sparksee::gdb::type_t t, sparksee::gdb::AttributeList &attrs, sparksee::gdb::Int32List &attrsPos\)](#)
Creates a new instance with the minimum common required arguments.
- [TypeLoader \(\)](#)
Creates a new instance.
- void [NotifyListeners \(const TypeLoaderEvent &ev\)](#)
Notifies progress to all registered listeners.
- void [SetHeadAttribute \(sparksee::gdb::attr_t attr\)](#)
Sets the attribute that will be used to find the head of the edge.
- void [SetHeadPosition \(sparksee::gdb::int32_t pos\)](#)
Sets the position of the head attribute in the source data.
- void [SetTailAttribute \(sparksee::gdb::attr_t attr\)](#)
Sets the attribute that will be used to find the tail of the edge.
- void [SetTailPosition \(sparksee::gdb::int32_t pos\)](#)
Sets the position of the tail attribute in the source data.

5.69.1 Detailed Description

Base [TypeLoader](#) class.

Base class to load a node or edge type from a graph using a [RowReader](#).

[TypeLoaderListener](#) can be registered to receive information about the progress of the load process by means of [TypeLoaderEvent](#). The default frequency of notification to listeners is 100000.

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.69.2 Member Enumeration Documentation**5.69.2.1 enum TypeLoader::Mode** [protected]

Load can work in different ways.

Enumerator:

ONE_PHASE Performs the load in a phases.

Load all objects an attributes at the same time.

TWO_PHASES Performs the load in two phases.

Firstly load all objects (and create them if necessary) and secondly loads all the attributes.

Working on this mode it is necessary to build a temporary file.

N_PHASES Performs the load in N phases.

Firstly load all objects (and create them if necessary) and secondly loads all the attributes. But in this case, attributes are loaded one by one. This way, if there are three attributes, then 4 traverses to the [RowReader](#) are necessary.

Working on this mode it is necessary to build a temporary file.

5.69.3 Constructor & Destructor Documentation

5.69.3.1 TypeLoader::TypeLoader (RowReader & *rr*, sparksee::gdb::Graph & *g*, sparksee::gdb::type_t *t*, sparksee::gdb::AttributeList & *attrs*, sparksee::gdb::Int32List & *attrsPos*) [protected]

Creates a new instance with the minimum common required arguments.

Parameters:

rr [in] Input [RowReader](#).

g [in] [Graph](#).

t [in] [Type](#) identifier.

attrs [in] [Attribute](#) identifiers to be loaded

attrsPos [in] [Attribute](#) positions (column index >=0)

5.69.4 Member Function Documentation

5.69.4.1 sparksee::gdb::bool_t TypeLoader::CanRun () [protected]

Checks that all the required settings are ready to run.

Returns:

Returns true if all the settings are ready.

5.69.4.2 `void TypeLoader::Run (Mode ph, sparksee::gdb::int32_t par) throw (sparksee::gdb::IOException, sparksee::gdb::Error) [protected]`

Runs load process.

Exceptions:

[IOException](#) If bad things happen reading from the [RowReader](#).

Parameters:

ph [in] The load mode.

par [in] Number of horizontal partitions to perform the load.

5.69.4.3 `void TypeLoader::NotifyListeners (const TypeLoaderEvent & ev) [protected]`

Notifies progress to all registered listeners.

Parameters:

ev [in] Progress event to be notified.

5.69.4.4 `void TypeLoader::SetHeadAttribute (sparksee::gdb::attr_t attr) [protected]`

Sets the attribute that will be used to find the head of the edge.

This method is protected because only the Edge loaders should have it.

Parameters:

attr [in] Head [Attribute](#)

Reimplemented in [EdgeTypeLoader](#).

5.69.4.5 `void TypeLoader::SetHeadPosition (sparksee::gdb::int32_t pos) [protected]`

Sets the position of the head attribute in the source data.

This method is protected because only the Edge loaders should have it.

Parameters:

pos [in] Head position

Reimplemented in [EdgeTypeLoader](#).

5.69.4.6 `void TypeLoader::SetTailAttribute (sparksee::gdb::attr_t attr) [protected]`

Sets the attribute that will be used to find the tail of the edge.

This method is protected because only the Edge loaders should have it.

Parameters:

attr [in] Tail [Attribute](#)

Reimplemented in [EdgeTypeLoader](#).

5.69.4.7 void TypeLoader::SetTailPosition (sparksee::gdb::int32_t pos) [protected]

Sets the position of the tail attribute in the source data.

This method is protected because only the Edge loaders should have it.

Parameters:

pos [in] Tail position

Reimplemented in [EdgeTypeLoader](#).

5.69.4.8 void TypeLoader::SetLogError (const std::wstring & path) throw (sparksee::gdb::IOException)

Sets a log error file.

By default errors are thrown as a exception and the load process ends. If a log file is set, errors are logged there and the load process does not stop.

Exceptions:

[IOException](#) If bad things happen opening the file.

5.69.4.9 void TypeLoader::SetLogOff ()

Turns off all the error reporting.

The log file will not be created and no exceptions for invalid data will be thrown. If you just want to turn off the logs, but abort at the first error what you should do is not call this method and not set a logError file.

5.69.4.10 void TypeLoader::Register (TypeLoaderListener & tel)

Registers a new listener.

Parameters:

← *tel* [TypeLoaderListener](#) to be registered.

5.69.4.11 virtual void TypeLoader::RunTwoPhases () throw (sparksee::gdb::IOException, sparksee::gdb::Error) [pure virtual]

Run the loader for two phases loading.

Firstly load all objects (and create them if necessary) and secondly loads all the attributes.

Working on this mode it is necessary to build a temporary file.

Implemented in [EdgeTypeLoader](#), and [NodeTypeLoader](#).

5.69.4.12 virtual void TypeLoader::RunNPhases (sparksee::gdb::int32_t partitions) throw (sparksee::gdb::IOException, sparksee::gdb::Error) [pure virtual]

Run the loader for N phases loading.

Firstly load all objects (and create them if necessary) and secondly loads all the attributes. But in this case, attributes are loaded one by one. This way, if there are three attributes, then 4 traverses are necessary.

Working on this mode it is necessary to build a temporary file.

Parameters:

partitions [in] Number of horizontal partitions to perform the load.

Implemented in [EdgeTypeLoader](#), and [NodeTypeLoader](#).

5.69.4.13 void TypeLoader::SetRowReader (RowReader & rr)

Sets the input data source.

Parameters:

rr [in] Input [RowReader](#).

5.69.4.14 void TypeLoader::SetGraph (sparksee::gdb::Graph & g)

Sets the graph where the data will be loaded.

Parameters:

g [in] [Graph](#).

5.69.4.15 void TypeLoader::SetLocale (const std::wstring & localeStr)

Sets the locale that will be used to read the data.

It should match the locale used in the rowreader.

Parameters:

localeStr [in] The locale string for the read data. See [CSVReader](#).

5.69.4.16 void TypeLoader::SetType (sparksee::gdb::type_t t)

Sets the type to be loaded.

Parameters:

t [in] [Type](#) identifier.

5.69.4.17 void TypeLoader::SetAttributes (sparksee::gdb::AttributeList & attrs)

Sets the list of Attributes.

Parameters:

attrs [in] [Attribute](#) identifiers to be loaded

5.69.4.18 void TypeLoader::SetAttributePositions (sparksee::gdb::Int32List & attrsPos)

Sets the list of attribute positions.

attrsPos [in] [Attribute](#) positions (column index >=0).

5.69.4.19 void TypeLoader::SetFrequency (sparksee::gdb::int32_t freq)

Sets the frequency of listener notification.

freq [in] Frequency in number of rows managed to notify progress to all listeners

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

- TypeLoader.h

5.70 TypeLoaderEvent Class Reference

Provides information about the progress of a [TypeLoader](#) instance.

Public Member Functions

- virtual [~TypeLoaderEvent](#) ()
Destructor.
- [sparksee::gdb::type_t GetTypeId](#) () const
Gets the type identifier.
- [sparksee::gdb::int64_t GetCount](#) () const
Gets the current number of objects created.
- [sparksee::gdb::int32_t GetPhase](#) () const
Gets the current phase.
- [sparksee::gdb::int32_t GetTotalPhases](#) () const
Gets the total number of phases.
- [sparksee::gdb::int32_t GetPartition](#) () const
Gets the current partition.
- [sparksee::gdb::int32_t GetTotalPartitions](#) () const
Gets the total number of partitions.
- [sparksee::gdb::int32_t GetTotalPartitionSteps](#) () const
Gets the total number of steps in the current partition.
- [sparksee::gdb::bool_t IsLast](#) () const
Gets if this is the last event or not.

Friends

- class [TypeLoader](#)

5.70.1 Detailed Description

Provides information about the progress of a [TypeLoader](#) instance.

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.70.2 Member Function Documentation

5.70.2.1 `sparksee::gdb::type_t TypeLoaderEvent::GetTypeId () const` `[inline]`

Gets the type identifier.

Returns:

The type identifier.

5.70.2.2 `sparksee::gdb::int64_t TypeLoaderEvent::GetCount () const` `[inline]`

Gets the current number of objects created.

Returns:

The current number of objects created.

5.70.2.3 `sparksee::gdb::int32_t TypeLoaderEvent::GetPhase () const` `[inline]`

Gets the current phase.

Returns:

The current phase.

5.70.2.4 `sparksee::gdb::int32_t TypeLoaderEvent::GetTotalPhases () const` `[inline]`

Gets the total number of phases.

Returns:

The total number of phases.

5.70.2.5 `sparksee::gdb::int32_t TypeLoaderEvent::GetPartition () const` `[inline]`

Gets the current partition.

Returns:

The current partition.

5.70.2.6 sparksee::gdb::int32_t TypeLoaderEvent::GetTotalPartitions () const [inline]

Gets the total number of partitions.

Returns:

The total number of partitions.

5.70.2.7 sparksee::gdb::int32_t TypeLoaderEvent::GetTotalPartitionSteps () const [inline]

Gets the total number of steps in the current partition.

Returns:

The total number steps in the current partition.

5.70.2.8 sparksee::gdb::bool_t TypeLoaderEvent::IsLast () const [inline]

Gets if this is the last event or not.

Returns:

TRUE if this is the last event, FALSE otherwise.

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

- TypeLoader.h

5.71 TypeLoaderListener Class Reference

Interface to be implemented to receive [TypeLoaderEvent](#) events from a [TypeLoader](#).

Public Member Functions

- virtual void [NotifyEvent](#) (const [TypeLoaderEvent](#) &ev)=0
Method to receive events from a [Loader](#).
- virtual [~TypeLoaderListener](#) ()
Destructor.

Protected Member Functions

- [TypeLoaderListener](#) ()
Protected because none should instantiate a [RowWriter](#).

5.71.1 Detailed Description

Interface to be implemented to receive [TypeLoaderEvent](#) events from a [TypeLoader](#).

Check out the 'Data import' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.71.2 Constructor & Destructor Documentation

5.71.2.1 TypeLoaderListener::TypeLoaderListener () [inline, protected]

Protected because none should instantiate a [RowWriter](#).

Just inherited classes may use this empty constructor.

5.71.3 Member Function Documentation

5.71.3.1 virtual void TypeLoaderListener::NotifyEvent (const TypeLoaderEvent & *ev*) [pure virtual]

Method to receive events from a [Loader](#).

Parameters:

ev [Loader.LoaderEvent](#) with information from a running [Loader](#).

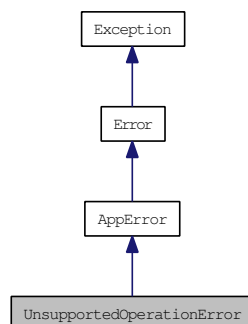
The documentation for this class was generated from the following file:

- TypeLoader.h

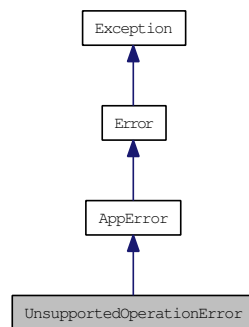
5.72 UnsupportedOperationException Class Reference

Unsupported operation error class.

Inheritance diagram for UnsupportedOperationException:



Collaboration diagram for UnsupportedOperationException:



Public Member Functions

- [UnsupportedOperationException](#) ()
Creates a new instance.
- [UnsupportedOperationException](#) (const std::string &mess)
Creates a new instance.
- virtual [~UnsupportedOperationException](#) ()
Destructor.
- const std::string & [Message](#) () const
Gets the message of the exception.
- void [SetMessage](#) (const std::string &mess)
Sets the message of the exception.

Static Public Member Functions

- static [Error NewError](#) (int32_t coreErrorCode)
Creates a new [Error](#) instance from a sparksee_core error code.

Protected Attributes

- std::string [message](#)
Message of the exception.

5.72.1 Detailed Description

Unsupported operation error class.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.72.2 Constructor & Destructor Documentation

5.72.2.1 `UnsupportedOperationException::UnsupportedOperationException (const std::string & mess)`

Creates a new instance.

Parameters:

mess [in] Message of the exception.

5.72.3 Member Function Documentation

5.72.3.1 `static Error Error::NewError (int32_t coreErrorCode) [static, inherited]`

Creates a new [Error](#) instance from a sparksee_core error code.

Parameters:

coreErrorCode [in] Sparkseecore error code.

Returns:

Depending on the given sparksee_core error, this may return an [Error](#) instance or an specific [Error](#) subclass instance.

5.72.3.2 `const std::string& Exception::Message () const [inherited]`

Gets the message of the exception.

It should be called GetMessage but this is not possible because of a macro defined in Windows.

Returns:

The message of the exception.

5.72.3.3 `void Exception::SetMessage (const std::string & mess) [inherited]`

Sets the message of the exception.

Parameters:

mess [in] Message.

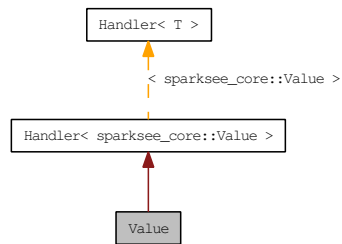
The documentation for this class was generated from the following file:

- Exception.h

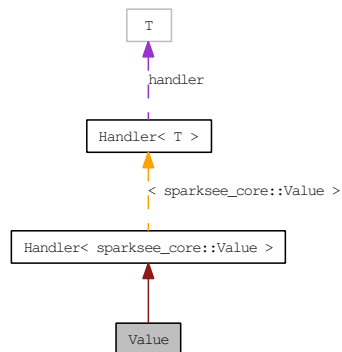
5.73 Value Class Reference

[Value](#) class.

Inheritance diagram for Value:



Collaboration diagram for Value:



Public Member Functions

- [Value \(\)](#)
Creates a new instance.
- [Value \(const Value &v\)](#)
Copy constructor.
- [virtual ~Value \(\)](#)
Destructor.
- [Value & operator= \(const Value &v\)](#)
Assignment operator.
- [bool_t IsNull \(\) const](#)
Gets if this is a NULL Value.
- [void SetNullVoid \(\)](#)
Sets the Value to NULL.
- [Value & SetNull \(\)](#)
Sets the Value to NULL.
- [DataType GetDataType \(\) const](#)

Gets the DataType.

- `bool_t GetBoolean () const`
Gets Boolean Value.
- `int32_t GetInteger () const`
Gets Integer Value.
- `int64_t GetLong () const`
Gets Long Value.
- `double64_t GetDouble () const`
Gets Double Value.
- `int64_t GetTimestamp () const`
Gets Timestamp Value.
- `const std::wstring & GetString () const`
Gets String Value.
- `oid_t GetOID () const`
Gets OID Value.
- `void SetBooleanVoid (bool_t v)`
Sets the Value.
- `Value & SetBoolean (bool_t v)`
Sets the Value.
- `void SetIntegerVoid (int32_t v)`
Sets the Value.
- `Value & SetInteger (int32_t v)`
Sets the Value.
- `void SetLongVoid (int64_t v)`
Sets the Value.
- `Value & SetLong (int64_t v)`
Sets the Value.
- `void SetDoubleVoid (double64_t v)`
Sets the Value.
- `Value & SetDouble (double64_t v)`
Sets the Value.
- `void SetTimestampVoid (int64_t v)`
Sets the Value.

- void [SetTimestampVoid](#) ([int32_t](#) year, [int32_t](#) month, [int32_t](#) day, [int32_t](#) hour, [int32_t](#) minutes, [int32_t](#) seconds, [int32_t](#) millisecs)
Sets the [Value](#).
- [Value](#) & [SetTimestamp](#) ([int64_t](#) v)
Sets the [Value](#).
- [Value](#) & [SetTimestamp](#) ([int32_t](#) year, [int32_t](#) month, [int32_t](#) day, [int32_t](#) hour, [int32_t](#) minutes, [int32_t](#) seconds, [int32_t](#) millisecs)
Sets the [Value](#).
- void [SetStringVoid](#) (const std::wstring &v)
Sets the [Value](#).
- [Value](#) & [SetString](#) (const std::wstring &v)
Sets the [Value](#).
- void [SetOIDVoid](#) ([oid_t](#) v)
Sets the [OID Value](#).
- [Value](#) & [SetOID](#) ([oid_t](#) v)
Sets the [Value](#).
- void [SetVoid](#) ([Value](#) &v)
Sets the [Value](#).
- [Value](#) & [Set](#) ([Value](#) &v)
Sets the [Value](#).
- [int32_t](#) [Compare](#) (const [Value](#) &v) const
Compares with the given [Value](#).
- [bool_t](#) [Equals](#) (const [Value](#) &v) const
Compares with the given [Value](#).
- std::wstring & [ToString](#) (std::wstring &str) const
Gets a string representation of the [Value](#).

Static Public Attributes

- static const [int32_t](#) [MaxLengthString](#)
Maximum number of characters allowed for a [String](#).

Friends

- class [Graph](#)
- class [ValuesIterator](#)

5.73.1 Detailed Description

[Value](#) class.

It is a container which stores a value and its data type (domain). A [Value](#) can be NULL.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.73.2 Constructor & Destructor Documentation

5.73.2.1 [Value::Value \(\)](#)

Creates a new instance.

It creates a NULL [Value](#).

5.73.2.2 [Value::Value \(const Value & v\)](#)

Copy constructor.

Parameters:

v [in] [Value](#) to be copied.

5.73.3 Member Function Documentation

5.73.3.1 [Value& Value::operator= \(const Value & v\)](#)

Assignment operator.

Parameters:

v [in] [Value](#) to be copied.

5.73.3.2 [bool_t Value::IsNull \(\) const](#)

Gets if this is a NULL [Value](#).

Returns:

TRUE if this is a NULL [Value](#), FALSE otherwise.

Reimplemented from [Handler< sparksee_core::Value >](#).

5.73.3.3 [Value& Value::SetNull \(\)](#) [inline]

Sets the [Value](#) to NULL.

Returns:

The calling instance.

5.73.3.4 `DataType Value::GetDataType () const`

Gets the `DataType`.

`Value` cannot be `NULL`.

Returns:

The `DataType`.

5.73.3.5 `bool_t Value::GetBoolean () const`

Gets Boolean `Value`.

This must be a non-`NULL` Boolean `Value`.

Returns:

The Boolean `Value`.

5.73.3.6 `int32_t Value::GetInteger () const`

Gets Integer `Value`.

This must be a non-`NULL` Integer `Value`.

Returns:

The Integer `Value`.

5.73.3.7 `int64_t Value::GetLong () const`

Gets Long `Value`.

This must be a non-`NULL` Long `Value`.

Returns:

The Long `Value`.

5.73.3.8 `double64_t Value::GetDouble () const`

Gets Double `Value`.

This must be a non-`NULL` Double `Value`.

Returns:

The Double `Value`.

5.73.3.9 `int64_t Value::GetTimestamp () const`

Gets Timestamp `Value`.

This must be a non-`NULL` Timestamp `Value`.

Returns:

The Timestamp `Value`.

5.73.3.10 `const std::wstring& Value::GetString () const`

Gets String [Value](#).

This must be a non-NULL String [Value](#).

Returns:

The String [Value](#).

5.73.3.11 `oid_t Value::GetOID () const`

Gets OID [Value](#).

This must be an non-NULL OID [Value](#).

Returns:

The OID [Value](#).

5.73.3.12 `void Value::SetBooleanVoid (bool_t v)`

Sets the [Value](#).

Parameters:

v [in] New Boolean value.

5.73.3.13 `Value& Value::SetBoolean (bool_t v) [inline]`

Sets the [Value](#).

Parameters:

v [in] Nex Boolean value.

Returns:

The calling instance.

5.73.3.14 `void Value::SetIntegerVoid (int32_t v)`

Sets the [Value](#).

Parameters:

v [in] New Integer value.

5.73.3.15 `Value& Value::SetInteger (int32_t v) [inline]`

Sets the [Value](#).

Parameters:

v [in] New Integer value.

Returns:

The calling instance.

5.73.3.16 void Value::SetLongVoid (int64_t v)

Sets the [Value](#).

Parameters:

v [in] New Long value.

5.73.3.17 Value& Value::SetLong (int64_t v) [inline]

Sets the [Value](#).

Parameters:

v [in] New Long value.

Returns:

The calling instance.

5.73.3.18 void Value::SetDoubleVoid (double64_t v)

Sets the [Value](#).

Parameters:

v [in] New Double value.

5.73.3.19 Value& Value::SetDouble (double64_t v) [inline]

Sets the [Value](#).

Parameters:

v [in] New Double value.

Returns:

The calling instance.

5.73.3.20 void Value::SetTimestampVoid (int64_t v)

Sets the [Value](#).

Parameters:

v [in] New Timestamp value.

5.73.3.21 `void Value::SetTimestampVoid (int32_t year, int32_t month, int32_t day, int32_t hour, int32_t minutes, int32_t seconds, int32_t millisecs)`

Sets the [Value](#).

Parameters:

year [in] The year (≥ 1970).
month [in] The month ([1..12]).
day [in] The of the month ([1..31]).
hour [in] The hour ([0..23]).
minutes [in] The minutes ([0..59]).
seconds [in] The seconds ([0..59]).
millisecs [in] The milliseconds ([0..999]).

5.73.3.22 `Value& Value::SetTimestamp (int64_t v) [inline]`

Sets the [Value](#).

Parameters:

v [in] New Timestamp value.

Returns:

The calling instance.

5.73.3.23 `Value& Value::SetTimestamp (int32_t year, int32_t month, int32_t day, int32_t hour, int32_t minutes, int32_t seconds, int32_t millisecs) [inline]`

Sets the [Value](#).

Parameters:

year [in] The year (≥ 1970).
month [in] The month ([1..12]).
day [in] The of the month ([1..31]).
hour [in] The hour ([0..23]).
minutes [in] The minutes ([0..59]).
seconds [in] The seconds ([0..59]).
millisecs [in] The milliseconds ([0..999]).

Returns:

The calling instance.

5.73.3.24 `void Value::SetStringVoid (const std::wstring & v)`

Sets the [Value](#).

Parameters:

v [in] New String value.

5.73.3.25 Value& Value::SetString (const std::wstring & *v*) `[inline]`

Sets the [Value](#).

Parameters:

v [in] New String value.

Returns:

The calling instance.

5.73.3.26 void Value::SetOIDVoid (oid_t *v*)

Sets the OID [Value](#).

Parameters:

v [in] New OID value.

5.73.3.27 Value& Value::SetOID (oid_t *v*) `[inline]`

Sets the [Value](#).

Parameters:

v [in] New OID [Value](#).

Returns:

The calling instance.

5.73.3.28 void Value::SetVoid (Value & *v*) `[inline]`

Sets the [Value](#).

Parameters:

v [in] New value.

5.73.3.29 Value& Value::Set (Value & *v*) `[inline]`

Sets the [Value](#).

Parameters:

v [in] New value.

Returns:

The calling instance.

5.73.3.30 int32_t Value::Compare (const Value & *v*) const

Compares with the given [Value](#).

It does not work for Text or if given [Value](#) objects does not have the same DataType.

Parameters:

v Given value to compare to.

Returns:

0 if this [Value](#) is equal to the given one; a value less than 0 if this [Value](#) is less than the given one; and a value greater than 0 if this [Value](#) is greater than the given one.

5.73.3.31 bool_t Value::Equals (const Value & *v*) const

Compares with the given [Value](#).

It does not work for Text or if given [Value](#) objects does not have the same DataType.

Parameters:

v Given value to compare to.

Returns:

TRUE if this [Value](#) is equal to the given one; FALSE otherwise.

5.73.3.32 std::wstring& Value::ToString (std::wstring & *str*) const

Gets a string representation of the [Value](#).

It does not work for Text.

Parameters:

str String to be used. It is cleared and set with the string representation of the [Value](#).

Returns:

The given string which has been updated.

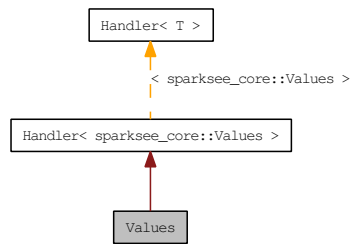
The documentation for this class was generated from the following file:

- [Value.h](#)

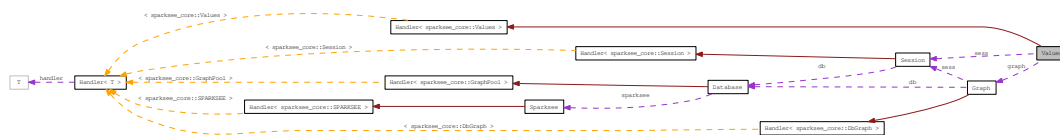
5.74 Values Class Reference

[Value](#) set class.

Inheritance diagram for Values:



Collaboration diagram for Values:



Public Member Functions

- virtual ~Values ()
Destructor.
- int64_t Count ()
Gets the number of elements into the collection.
- ValuesIterator * Iterator (Order order)
Gets a ValuesIterator.

Friends

- class **Graph**
- class **ValuesIterator**

5.74.1 Detailed Description

Value set class.

This is a set of **Value** instances, that is there is no duplicated elements.

Use a [ValuesIterator](#) to traverse all the elements into the set.

When the `Values` instance is closed, it closes all existing and non-closed `ValuesIterator` instances too.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.74.2 Member Function Documentation

5.74.2.1 int64_t Values::Count ()

Gets the number of elements into the collection.

Returns:

The number of elements into the collection.

5.74.2.2 ValuesIterator* Values::Iterator (Order *order*)

Gets a [ValuesIterator](#).

Returns:

[ValuesIterator](#) instance.

Parameters:

order [in] Ascending or descending order.

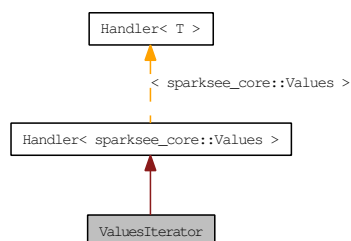
The documentation for this class was generated from the following file:

- [Values.h](#)

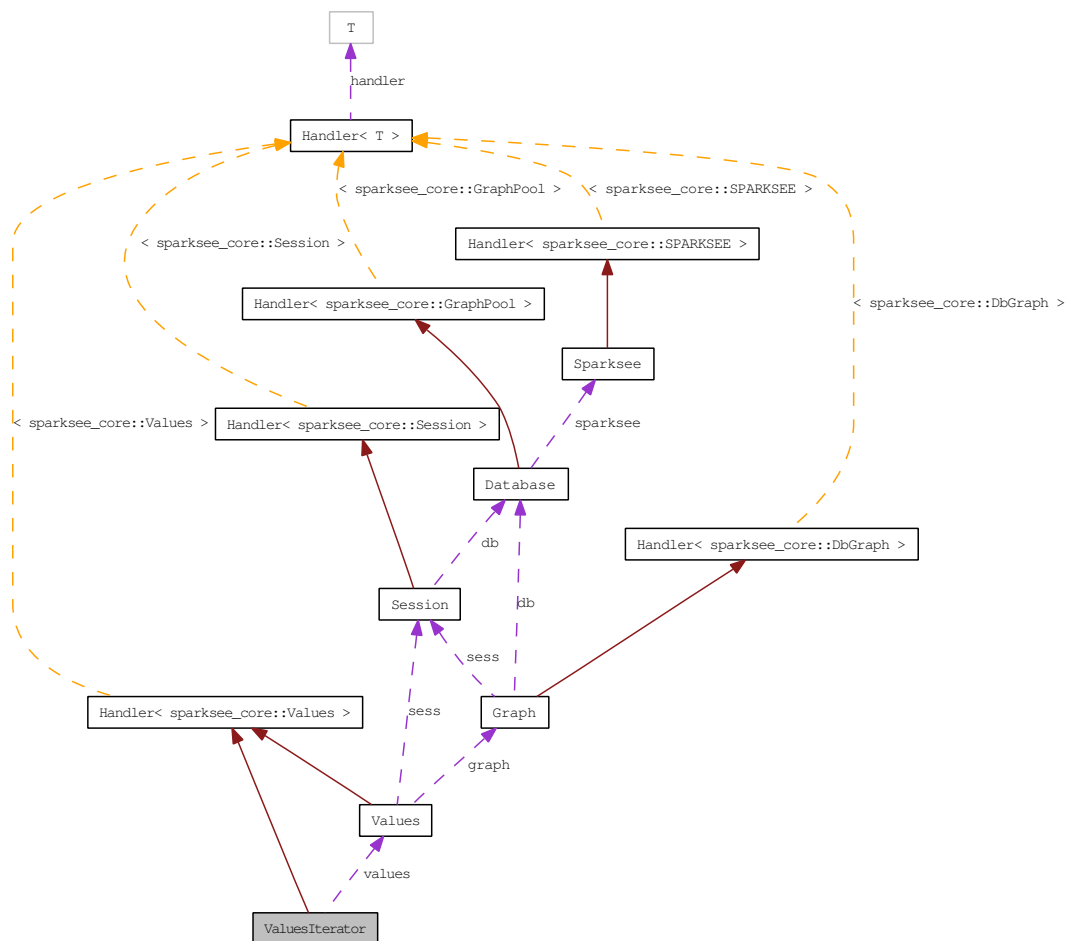
5.75 ValuesIterator Class Reference

[Values](#) iterator class.

Inheritance diagram for ValuesIterator:



Collaboration diagram for ValuesIterator:



Public Member Functions

- virtual `~ValuesIterator()`
Destructor.
- `bool_t HasNext()`
Gets if there are more elements to traverse.
- `Value * Next()`
Gets the next element to traverse.

Friends

- class `Values`

5.75.1 Detailed Description

[Values](#) iterator class.

It allows for traversing all the elements into a [Values](#) instance.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.75.2 Member Function Documentation

5.75.2.1 `bool_t ValuesIterator::HasNext ()`

Gets if there are more elements to traverse.

Returns:

TRUE if there are more elements to traverse, FALSE otherwise.

5.75.2.2 `Value* ValuesIterator::Next ()`

Gets the next element to traverse.

Returns:

The next element.

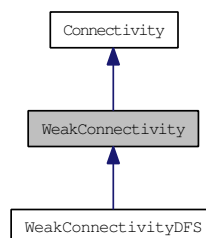
The documentation for this class was generated from the following file:

- ValuesIterator.h

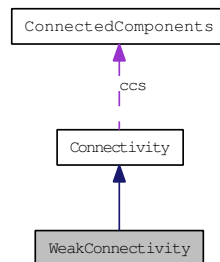
5.76 WeakConnectivity Class Reference

[WeakConnectivity](#) class.

Inheritance diagram for WeakConnectivity:



Collaboration diagram for WeakConnectivity:



Public Member Functions

- virtual [~WeakConnectivity](#) ()
Destructor.
- virtual void [AddEdgeType](#) (sparksee::gdb::type_t t)
Allows connectivity through edges of the given type.
- virtual void [AddAllEdgeTypes](#) ()
Allows connectivity through all edge types of the graph.
- virtual void [AddNodeType](#) (sparksee::gdb::type_t t)
Allows connectivity through nodes of the given type.
- virtual void [AddAllNodeTypes](#) ()
Allows connectivity through all node types of the graph.
- virtual void [ExcludeNodes](#) (sparksee::gdb::Objects &nodes)
Set which nodes can't be used.
- virtual void [ExcludeEdges](#) (sparksee::gdb::Objects &edges)
Set which edges can't be used.
- [ConnectedComponents](#) * [GetConnectedComponents](#) ()
Returns the results generated by the execution of the algorithm.
- virtual void [Run](#) ()=0
Runs the algorithm in order to find the connected components.
- void [SetMaterializedAttribute](#) (const std::wstring &attributeName)
Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Protected Types

- typedef std::map< [sparksee::gdb::type_t](#), [sparksee::gdb::EdgesDirection](#) > [EdgeTypes_t](#)
A type definition to store allowed edge types.

- typedef std::vector< [sparksee::gdb::type_t](#) > [NodeTypes_t](#)
A type definition to store allowed node types.

Protected Member Functions

- [WeakConnectivity](#) ([sparksee::gdb::Session](#) &s)
Creates a new instance of [WeakConnectivity](#).
- void [AddEdgeType](#) ([sparksee::gdb::type_t](#) t, [sparksee::gdb::EdgesDirection](#) d)
Allows connectivity through edges of the given type.
- void [AddAllEdgeTypes](#) ([sparksee::gdb::EdgesDirection](#) d)
Allows connectivity through all edge types of the graph.
- void [AssertAddedEdges](#) ()
Check that edges had been added.
- void [AssertAddedNodes](#) ()
Check that nodes had been added.
- void [AssertNotComputed](#) ()
Check that the connectivity had not been calculated yet.
- void [SetConnectedComponent](#) ([sparksee::gdb::oid_t](#) idNode)
Assigns the current component to the given node.
- void [SetNodesNotVisited](#) ()
Set all the selected nodes in [nodesNotVisited](#).
- void [AssertNotComponentAttribute](#) (const std::wstring &attributeName)
Check that the given attribute name is not already in use.
- void [AssertComputed](#) ()
Check that the connectivity had been calculated.
- void [AssertEdgeType](#) ([sparksee::gdb::type_t](#) edgetype)
Check that the given edge type is valid.
- void [AssertNodeType](#) ([sparksee::gdb::type_t](#) nodetype)
Check that the given node type is valid.
- void [CreateGlobalPersistentAttribute](#) (const std::wstring &attributeName)
Set a new persistent global attribute to store the connectivity information.
- void [CreateGlobalTransientAttribute](#) ()
Set a new temporary global attribute to store the connectivity information.
- void [RemoveGlobalAttribute](#) ()

Removes the global attribute where the connectivity information is stored.

- `sparksee::gdb::bool_t IsNodeTypeAllowed (sparksee::gdb::oid_t nodeId)`
Check if the given node has an allowed type.
- `sparksee::gdb::bool_t IsNodeExcluded (sparksee::gdb::oid_t node)`
Check if the given node is forbidden.
- `sparksee::gdb::bool_t IsEdgeExcluded (sparksee::gdb::oid_t edge)`
Check if the given edge is forbidden.

Protected Attributes

- `sparksee::gdb::Session * sess`
Session.
- `sparksee::gdb::Graph * graph`
Graph.
- `EdgeTypes_t edgeTypes`
Allowed edge types.
- `std::vector< sparksee::gdb::type_t > nodeTypes`
Allowed node types.
- `sparksee::gdb::attr_t attrComponent`
common attribute where the connected component information is stored.
- `std::wstring attrComponentName`
name of the common attribute where the connected component information is stored.
- `sparksee::gdb::int64_t actualComponent`
Current component identifier.
- `sparksee::gdb::Objects * nodesNotVisited`
Identifiers of the nodes not visited.
- `sparksee::gdb::bool_t matResults`
Materialized results.
- `sparksee::gdb::bool_t computed`
True if the connectivity has been calculated.
- `sparksee::gdb::Objects * excludedNodes`
The set of excluded nodes.
- `sparksee::gdb::Objects * excludedEdges`
The set of excluded edges.

- [ConnectedComponents](#) * [ccs](#)

The calculated connectivity information.

5.76.1 Detailed Description

[WeakConnectivity](#) class.

Any class implementing this abstract class can be used to solve the problem of finding weakly connected components in an **undirected** graph or in a directed graph which will be considered as an undirected one.

It consists in finding components where every pair (u,v) of nodes contained in it has a path from u to v and from v to u.

It is possible to set some restrictions after constructing a new instance of this class and before running it in order to limit the results.

After the execution, we can retrieve the results stored in an instance of the [ConnectedComponents](#) class using the `getConnectedComponents` method.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.76.2 Constructor & Destructor Documentation

5.76.2.1 `WeakConnectivity::WeakConnectivity (sparksee::gdb::Session & s)` [protected]

Creates a new instance of [WeakConnectivity](#).

Parameters:

s [in] [Session](#) to get the graph from and calculate the connectivity

5.76.3 Member Function Documentation

5.76.3.1 `virtual void WeakConnectivity::AddEdgeType (sparksee::gdb::type_t t)` [virtual]

Allows connectivity through edges of the given type.

In a weak connectivity the edges can be used in [Any](#) direction.

Parameters:

t [in] Edge type.

d [in] Edge direction.

5.76.3.2 `virtual void WeakConnectivity::AddAllEdgeTypes ()` [virtual]

Allows connectivity through all edge types of the graph.

In a weak connectivity the edges can be used in [Any](#) direction.

Parameters:

d [in] Edge direction.

5.76.3.3 virtual void Connectivity::ExcludeNodes (sparksee::gdb::Objects & *nodes*) [virtual, inherited]

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters:

nodes [in] A set of node identifiers that must be kept intact until the destruction of the class.

5.76.3.4 virtual void Connectivity::ExcludeEdges (sparksee::gdb::Objects & *edges*) [virtual, inherited]

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters:

edges [in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.76.3.5 ConnectedComponents* Connectivity::GetConnectedComponents () [inherited]

Returns the results generated by the execution of the algorithm.

These results contain information related to the connected components found as the number of different components, the set of nodes contained in each component or many other data.

Returns:

Returns an instance of the class [ConnectedComponents](#) which contain information related to the connected components found.

5.76.3.6 virtual void Connectivity::Run () [pure virtual, inherited]

Runs the algorithm in order to find the connected components.

This method can be called only once.

Implemented in [StrongConnectivityGabow](#), and [WeakConnectivityDFS](#).

5.76.3.7 void Connectivity::SetMaterializedAttribute (const std::wstring & *attributeName*) [inherited]

Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it is only necessary to create a new instance of the class [ConnectedComponents](#) indicating the graph and the name of the common attribute type which stores the results. This instance will have all the information related to the connected components found in the moment of the execution of the algorithm that stored this data.

It is possible to run the algorithm without specifying this parameter in order to avoid materializing the results of the execution.

Parameters:

attributeName [in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

5.76.3.8 void **Connectivity::AddEdgeType** (sparksee::gdb::type_t *t*, sparksee::gdb::EdgesDirection *d*) [protected, inherited]

Allows connectivity through edges of the given type.

Parameters:

t [in] Edge type.

d [in] Edge direction.

Reimplemented in [StrongConnectivity](#).

5.76.3.9 void **Connectivity::AddAllEdgeTypes** (sparksee::gdb::EdgesDirection *d*) [protected, inherited]

Allows connectivity through all edge types of the graph.

Parameters:

d [in] Edge direction.

Reimplemented in [StrongConnectivity](#).

5.76.3.10 void **Connectivity::SetNodesNotVisited** () [protected, inherited]

Set all the selected nodes in nodesNotVisited.

That's all the nodes of the allowed node types but not the excluded ones.

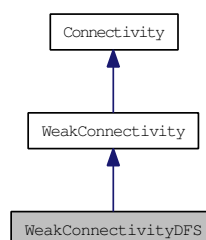
The documentation for this class was generated from the following file:

- WeakConnectivity.h

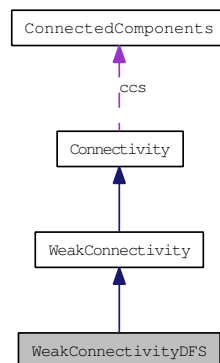
5.77 WeakConnectivityDFS Class Reference

[WeakConnectivityDFS](#) class.

Inheritance diagram for WeakConnectivityDFS:



Collaboration diagram for WeakConnectivityDFS:



Public Member Functions

- [WeakConnectivityDFS](#) (sparksee::gdb::Session &s)
Creates a new instance of [WeakConnectivityDFS](#).
- virtual [~WeakConnectivityDFS](#) ()
Destructor.
- void [Run](#) ()
Executes the algorithm.
- virtual void [AddEdgeType](#) (sparksee::gdb::type_t t)
Allows connectivity through edges of the given type.
- virtual void [AddAllEdgeTypes](#) ()
Allows connectivity through all edge types of the graph.
- virtual void [AddNodeType](#) (sparksee::gdb::type_t t)
Allows connectivity through nodes of the given type.
- virtual void [AddAllNodeTypes](#) ()
Allows connectivity through all node types of the graph.
- virtual void [ExcludeNodes](#) (sparksee::gdb::Objects &nodes)
Set which nodes can't be used.
- virtual void [ExcludeEdges](#) (sparksee::gdb::Objects &edges)
Set which edges can't be used.
- [ConnectedComponents](#) * [GetConnectedComponents](#) ()
Returns the results generated by the execution of the algorithm.
- void [SetMaterializedAttribute](#) (const std::wstring &attributeName)
Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Protected Types

- typedef std::map< [sparksee::gdb::type_t](#), [sparksee::gdb::EdgesDirection](#) > [EdgeTypes_t](#)
A type definition to store allowed edge types.
- typedef std::vector< [sparksee::gdb::type_t](#) > [NodeTypes_t](#)
A type definition to store allowed node types.

Protected Member Functions

- void [AddEdgeType](#) ([sparksee::gdb::type_t](#) t, [sparksee::gdb::EdgesDirection](#) d)
Allows connectivity through edges of the given type.
- void [AddAllEdgeTypes](#) ([sparksee::gdb::EdgesDirection](#) d)
Allows connectivity through all edge types of the graph.
- void [AssertAddedEdges](#) ()
Check that edges had been added.
- void [AssertAddedNodes](#) ()
Check that nodes had been added.
- void [AssertNotComputed](#) ()
Check that the connectivity had not been calculated yet.
- void [SetConnectedComponent](#) ([sparksee::gdb::oid_t](#) idNode)
Assigns the current component to the given node.
- void [SetNodesNotVisited](#) ()
Set all the selected nodes in nodesNotVisited.
- void [AssertNotComponentAttribute](#) (const std::wstring &attributeName)
Check that the given attribute name is not already in use.
- void [AssertComputed](#) ()
Check that the connectivity had been calculated.
- void [AssertEdgeType](#) ([sparksee::gdb::type_t](#) edgetype)
Check that the given edge type is valid.
- void [AssertNodeType](#) ([sparksee::gdb::type_t](#) nodetype)
Check that the given node type is valid.
- void [CreateGlobalPersistentAttribute](#) (const std::wstring &attributeName)
Set a new persistent global attribute to store the connectivity information.
- void [CreateGlobalTransientAttribute](#) ()
Set a new temporary global attribute to store the connectivity information.

- void [RemoveGlobalAttribute](#) ()
Removes the global attribute where the connectivity information is stored.
- [sparksee::gdb::bool_t IsNodeTypeAllowed](#) ([sparksee::gdb::oid_t](#) nodeId)
Check if the given node has an allowed type.
- [sparksee::gdb::bool_t IsNodeExcluded](#) ([sparksee::gdb::oid_t](#) node)
Check if the given node is forbidden.
- [sparksee::gdb::bool_t IsEdgeExcluded](#) ([sparksee::gdb::oid_t](#) edge)
Check if the given edge is forbidden.

Protected Attributes

- [sparksee::gdb::Session * sess](#)
Session.
- [sparksee::gdb::Graph * graph](#)
Graph.
- [EdgeTypes_t edgeTypes](#)
Allowed edge types.
- [std::vector< sparksee::gdb::type_t > nodeTypes](#)
Allowed node types.
- [sparksee::gdb::attr_t attrComponent](#)
common attribute where the connected component information is stored.
- [std::wstring attrComponentName](#)
name of the common attribute where the connected component information is stored.
- [sparksee::gdb::int64_t actualComponent](#)
Current component identifier.
- [sparksee::gdb::Objects * nodesNotVisited](#)
Identifiers of the nodes not visited.
- [sparksee::gdb::bool_t matResults](#)
Materialized results.
- [sparksee::gdb::bool_t computed](#)
True if the connectivity has been calculated.
- [sparksee::gdb::Objects * excludedNodes](#)
The set of excluded nodes.
- [sparksee::gdb::Objects * excludedEdges](#)

The set of excluded edges.

- [ConnectedComponents](#) * `ccs`

The calculated connectivity information.

5.77.1 Detailed Description

[WeakConnectivityDFS](#) class.

This class can be used to solve the problem of finding weakly connected components in an **undirected** graph or in a directed graph which will be considered as an undirected one.

It consists in finding components where every pair (u,v) of nodes contained in it has a path from u to v and from v to u. This implementation is based on the Depth-First Search (DFS) algorithm.

It is possible to set some restrictions after constructing a new instance of this class and before running it in order to limit the results.

After the execution, we can retrieve the results stored in an instance of the [ConnectedComponents](#) class using the `getConnectedComponents` method.

Check out the 'Algorithms' section in the SPARKSEE User Manual for more details on this.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.77.2 Constructor & Destructor Documentation

5.77.2.1 WeakConnectivityDFS::WeakConnectivityDFS (sparksee::gdb::Session & s)

Creates a new instance of [WeakConnectivityDFS](#).

After creating this instance is required to indicate the set of edge types and the set of node types which will be navigated through while traversing the graph in order to find the weak connected components.

Parameters:

`s` [in] [Session](#) to get the graph from and calculate the connectivity

5.77.3 Member Function Documentation

5.77.3.1 virtual void WeakConnectivity::AddEdgeType (sparksee::gdb::type_t t) [virtual, inherited]

Allows connectivity through edges of the given type.

In a weak connectivity the edges can be used in [Any](#) direction.

Parameters:

`t` [in] Edge type.

`d` [in] Edge direction.

5.77.3.2 `void Connectivity::AddEdgeType (sparksee::gdb::type_t t, sparksee::gdb::EdgesDirection d)` [protected, inherited]

Allows connectivity through edges of the given type.

Parameters:

t [in] Edge type.

d [in] Edge direction.

Reimplemented in [StrongConnectivity](#).

5.77.3.3 `virtual void WeakConnectivity::AddAllEdgeTypes ()` [virtual, inherited]

Allows connectivity through all edge types of the graph.

In a weak connectivity the edges can be used in [Any](#) direction.

Parameters:

d [in] Edge direction.

5.77.3.4 `void Connectivity::AddAllEdgeTypes (sparksee::gdb::EdgesDirection d)` [protected, inherited]

Allows connectivity through all edge types of the graph.

Parameters:

d [in] Edge direction.

Reimplemented in [StrongConnectivity](#).

5.77.3.5 `virtual void Connectivity::ExcludeNodes (sparksee::gdb::Objects & nodes)` [virtual, inherited]

Set which nodes can't be used.

This will replace any previously specified set of excluded nodes. Should only be used to exclude the usage of specific nodes from allowed node types because it's less efficient than not allowing a node type.

Parameters:

nodes [in] A set of node identifiers that must be kept intact until the destruction of the class.

5.77.3.6 `virtual void Connectivity::ExcludeEdges (sparksee::gdb::Objects & edges)` [virtual, inherited]

Set which edges can't be used.

This will replace any previously specified set of excluded edges. Should only be used to exclude the usage of specific edges from allowed edge types because it's less efficient than not allowing an edge type.

Parameters:

edges [in] A set of edge identifiers that must be kept intact until the destruction of the class.

5.77.3.7 ConnectedComponents* Connectivity::GetConnectedComponents () [inherited]

Returns the results generated by the execution of the algorithm.

These results contain information related to the connected components found as the number of different components, the set of nodes contained in each component or many other data.

Returns:

Returns an instance of the class [ConnectedComponents](#) which contain information related to the connected components found.

5.77.3.8 void Connectivity::SetMaterializedAttribute (const std::wstring & attributeName) [inherited]

Creates a new common attribute type for all node types in the graph in order to store, persistently, the results related to the connected components found while executing this algorithm.

Whenever the user wants to retrieve the results, even when the graph has been closed and opened again, it is only necessary to create a new instance of the class [ConnectedComponents](#) indicating the graph and the name of the common attribute type which stores the results. This instance will have all the information related to the connected components found in the moment of the execution of the algorithm that stored this data.

It is possible to run the algorithm without specifying this parameter in order to avoid materializing the results of the execution.

Parameters:

attributeName [in] The name of the common attribute type for all node types in the graph which will store persistently the results generated by the execution of the algorithm.

5.77.3.9 void Connectivity::SetNodesNotVisited () [protected, inherited]

Set all the selected nodes in nodesNotVisited.

That's all the nodes of the allowed node types but not the excluded ones.

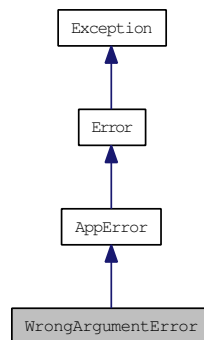
The documentation for this class was generated from the following file:

- WeakConnectivityDFS.h

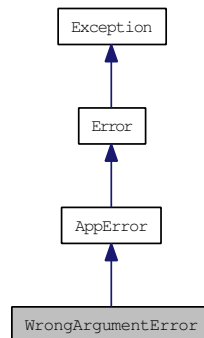
5.78 WrongArgumentError Class Reference

Wrong argument error class.

Inheritance diagram for WrongArgumentError:



Collaboration diagram for WrongArgumentError:



Public Member Functions

- [WrongArgumentError \(\)](#)
Creates a new instance.
- [WrongArgumentError \(const std::string &mess\)](#)
Creates a new instance.
- virtual [~WrongArgumentError \(\)](#)
Destructor.
- const std::string & [Message \(\)](#) const
Gets the message of the exception.
- void [SetMessage \(const std::string &mess\)](#)
Sets the message of the exception.

Static Public Member Functions

- static [Error NewError \(int32_t coreErrorCode\)](#)

Creates a new [Error](#) instance from a sparksee_core error code.

Protected Attributes

- `std::string message`
Message of the exception.

5.78.1 Detailed Description

Wrong argument error class.

Author:

Sparsity Technologies <http://www.sparsity-technologies.com>

5.78.2 Constructor & Destructor Documentation

5.78.2.1 WrongArgumentError::WrongArgumentError (const std::string & mess)

Creates a new instance.

Parameters:

mess [in] Message of the exception.

5.78.3 Member Function Documentation

5.78.3.1 static `Error Error::NewError (int32_t coreErrorCode)` [static, inherited]

Creates a new [Error](#) instance from a sparksee_core error code.

Parameters:

coreErrorCode [in] Sparkseecore error code.

Returns:

Depending on the given sparksee_core error, this may return an [Error](#) instance or an specific [Error](#) subclass instance.

5.78.3.2 `const std::string& Exception::Message () const` [inherited]

Gets the message of the exception.

It should be called GetMessage but this is not possible because of a macro defined in Windows.

Returns:

The message of the exception.

5.78.3.3 void Exception::SetMessage (const std::string & *mess*) [inherited]

Sets the message of the exception.

Parameters:

mess [in] Message.

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

- Exception.h

Index

- ~Handler
 - [gdb](#), [20](#)
- Add
 - [AttributeList](#), [31](#)
 - [BooleanList](#), [37](#)
 - [Int32List](#), [114](#)
 - [Objects](#), [142](#)
 - [OIDList](#), [150](#)
 - [StringList](#), [204](#)
 - [TypeList](#), [246](#)
- AddAllEdgeTypes
 - [Connectivity](#), [46](#)
 - [Context](#), [48](#)
 - [ShortestPath](#), [165](#)
 - [SinglePairShortestPath](#), [171](#)
 - [SinglePairShortestPathBFS](#), [176](#)
 - [SinglePairShortestPathDijkstra](#), [183](#)
 - [StrongConnectivity](#), [209](#)
 - [StrongConnectivityGabow](#), [215](#)
 - [Traversal](#), [224](#)
 - [TraversalBFS](#), [229](#)
 - [TraversalDFS](#), [233](#)
 - [WeakConnectivity](#), [277](#), [279](#)
 - [WeakConnectivityDFS](#), [284](#)
- AddEdgeType
 - [Connectivity](#), [46](#)
 - [Context](#), [48](#)
 - [ShortestPath](#), [165](#)
 - [SinglePairShortestPath](#), [171](#)
 - [SinglePairShortestPathBFS](#), [176](#)
 - [SinglePairShortestPathDijkstra](#), [182](#)
 - [StrongConnectivity](#), [209](#)
 - [StrongConnectivityGabow](#), [214](#)
 - [Traversal](#), [224](#)
 - [TraversalBFS](#), [229](#)
 - [TraversalDFS](#), [233](#)
 - [WeakConnectivity](#), [277](#), [279](#)
 - [WeakConnectivityDFS](#), [283](#)
- AddWeightedEdgeType
 - [SinglePairShortestPathDijkstra](#), [182](#)
- [Algorithms](#), [23](#)
- Any
 - [gdb](#), [17](#)
 - [Objects](#), [142](#)
- [AppError](#), [25](#)
 - [AppError](#), [26](#)
 - [Message](#), [27](#)
 - [NewError](#), [26](#)
 - [SetMessage](#), [27](#)
- [Ascendent](#)
 - [gdb](#), [19](#)
- [AsDirected](#)
 - [EdgeExport](#), [67](#)
- [Attribute](#), [27](#)
 - [GetCount](#), [29](#)
 - [GetDataType](#), [29](#)
 - [GetId](#), [28](#)
 - [GetKind](#), [29](#)
 - [GetName](#), [28](#)
 - [GetSize](#), [29](#)
 - [GetTypeId](#), [28](#)
 - [IsSessionAttribute](#), [29](#)
- [AttributeKind](#)
 - [gdb](#), [15](#)
- [AttributeList](#), [30](#)
 - [Add](#), [31](#)
 - [AttributeList](#), [30](#)
 - [Count](#), [31](#)
 - [Iterator](#), [31](#)
- [AttributeListIterator](#), [31](#)
 - [HasNext](#), [32](#)
 - [Next](#), [32](#)
- [AttributeStatistics](#), [32](#)
 - [GetAvgLengthString](#), [35](#)
 - [GetDistinct](#), [34](#)
 - [GetMax](#), [34](#)
 - [GetMaxLengthString](#), [34](#)
 - [GetMean](#), [35](#)
 - [GetMedian](#), [36](#)
 - [GetMin](#), [34](#)
 - [GetMinLengthString](#), [35](#)
 - [GetMode](#), [35](#)
 - [GetModeCount](#), [35](#)
 - [GetNull](#), [34](#)
 - [GetTotal](#), [34](#)
 - [GetVariance](#), [35](#)
- [Backup](#)
 - [Graph](#), [109](#)
- [Basic](#)
 - [gdb](#), [15](#)
- [Between](#)
 - [gdb](#), [16](#)
- [Boolean](#)
 - [gdb](#), [17](#)
- [BooleanList](#), [36](#)
 - [Add](#), [37](#)
 - [BooleanList](#), [37](#)
 - [Count](#), [37](#)
 - [Iterator](#), [37](#)
- [BooleanListIterator](#), [38](#)

- HasNext, 38
- Next, 38
- Box
 - gdb, 18
- CanRun
 - EdgeTypeExporter, 72
 - EdgeTypeLoader, 78
 - NodeTypeExporter, 128
 - NodeTypeLoader, 134
 - TypeExporter, 239
 - TypeLoader, 250
- CheckOnlyExistence
 - SinglePairShortestPathBFS, 176
- Close
 - CSVReader, 54
 - CSVWriter, 57
 - RowReader, 155
 - RowWriter, 157
 - TextStream, 221
- ColorRGB
 - gdb, 15
- CombineDifference
 - Objects, 144
- CombineIntersection
 - Objects, 144
- CombineUnion
 - Objects, 144
- Compare
 - Value, 268
- Compute
 - Context, 49
- Condition
 - gdb, 15
- Config
 - gdb, 18
- ConnectedComponents, 39
 - ConnectedComponents, 39
 - GetConnectedComponent, 40
 - GetCount, 40
 - GetNodes, 40
 - GetSize, 40
- Connectivity, 41
 - AddAllEdgeTypes, 46
 - AddEdgeType, 46
 - Connectivity, 44
 - ExcludeEdges, 45
 - ExcludeNodes, 45
 - GetConnectedComponents, 45
 - Run, 45
 - SetMaterializedAttribute, 45
 - SetNodesNotVisited, 46
- Contains
 - Objects, 144
- Context, 46
 - AddAllEdgeTypes, 48
 - AddEdgeType, 48
 - Compute, 49
 - Context, 48
 - ExcludeEdges, 49
 - ExcludeNodes, 49
 - SetMaximumHops, 49
- Copy
 - Objects, 141, 145
- Count
 - AttributeList, 31
 - BooleanList, 37
 - Int32List, 114
 - Objects, 142
 - OIDList, 149
 - StringList, 203
 - TypeList, 246
 - Values, 271
- CountEdges
 - Graph, 96
- CountNodes
 - Graph, 96
- Create
 - Sparksee, 185
- CSVReader, 50
 - Close, 54
 - GetRow, 54
 - Open, 53
 - Read, 53
 - Reset, 53
 - SetLocale, 53
 - SetMultilines, 52
 - SetNumLines, 53
 - SetQuotes, 52
 - SetSeparator, 52
 - SetStartLine, 52
- CSVWriter, 54
 - Close, 57
 - Open, 57
 - SetAutoQuotes, 56
 - SetForcedQuotes, 56
 - SetLocale, 56
 - SetQuotes, 56
 - SetSeparator, 56
 - Write, 57
- Database, 58
 - GetAlias, 59
 - GetPath, 59
 - GetStatistics, 59
- DatabaseStatistics, 60
 - GetCache, 61
 - GetData, 61

- GetRead, 61
- GetSessions, 61
- GetTemp, 61
- GetWrite, 61
- DataType
 - gdb, 16
- Debug
 - gdb, 18
- DefaultExport, 62
 - EnableType, 64
 - GetEdge, 64
 - GetEdgeType, 63
 - GetGraph, 63
 - GetNode, 64
 - GetNodeType, 63
- Degree
 - Graph, 106
- Descendent
 - gdb, 19
- Difference
 - Objects, 143
- Double
 - gdb, 17
- Drop
 - Graph, 97
- DumpData
 - Graph, 109
- DumpStorage
 - Graph, 109
- Edge
 - gdb, 18
- EdgeData, 64
 - GetEdge, 65
 - GetHead, 65
 - GetTail, 65
- EdgeExport, 66
 - AsDirected, 67
 - GetColorRGB, 68
 - GetFontSize, 69
 - GetLabel, 67
 - GetLabelColorRGB, 68
 - GetWidth, 68
 - SetAsDirected, 67
 - SetColorRGB, 68
 - SetFontSize, 69
 - SetLabel, 67
 - SetLabelColorRGB, 68
 - SetWidth, 68
- Edges
 - Graph, 106
- EdgesDirection
 - gdb, 17
- EdgeTypeExporter, 69
 - CanRun, 72
 - EdgeTypeExporter, 71
 - NotifyListeners, 72
 - Register, 72
 - RunProcess, 72
 - SetAttributes, 73
 - SetFrequency, 73
 - SetGraph, 73
 - SetHeadAttribute, 71
 - SetHeader, 73
 - SetHeadPosition, 71
 - SetRowWriter, 73
 - SetTailAttribute, 72
 - SetTailPosition, 72
 - SetType, 73
- EdgeTypeLoader, 74
 - CanRun, 78
 - EdgeTypeLoader, 77
 - Mode, 76
 - N_PHASES, 76
 - NotifyListeners, 78
 - ONE_PHASE, 76
 - Register, 79
 - Run, 78
 - SetAttributePositions, 80
 - SetAttributes, 80
 - SetFrequency, 80
 - SetGraph, 79
 - SetHeadAttribute, 77
 - SetHeadPosition, 77
 - SetLocale, 79
 - SetLogError, 78
 - SetLogOff, 79
 - SetRowReader, 79
 - SetTailAttribute, 77
 - SetTailPosition, 78
 - SetType, 79
 - TWO_PHASES, 76
- EnableType
 - DefaultExport, 64
 - ExportManager, 87
- Equal
 - gdb, 16
- Equals
 - Objects, 143
 - Value, 269
- Error, 80
 - Error, 82
 - Message, 82
 - NewError, 82
 - SetMessage, 82
- Exception, 82
 - Exception, 84
 - Message, 84

- SetMessage, 84
- ExcludeEdges
 - Connectivity, 45
 - Context, 49
 - ShortestPath, 166
 - SinglePairShortestPath, 171
 - SinglePairShortestPathBFS, 177
 - SinglePairShortestPathDijkstra, 183
 - StrongConnectivity, 209
 - StrongConnectivityGabow, 215
 - Traversal, 224
 - TraversalBFS, 229
 - TraversalDFS, 234
 - WeakConnectivity, 278
 - WeakConnectivityDFS, 284
- ExcludeNodes
 - Connectivity, 45
 - Context, 49
 - ShortestPath, 166
 - SinglePairShortestPath, 171
 - SinglePairShortestPathBFS, 177
 - SinglePairShortestPathDijkstra, 183
 - StrongConnectivity, 209
 - StrongConnectivityGabow, 215
 - Traversal, 224
 - TraversalBFS, 229
 - TraversalDFS, 233
 - WeakConnectivity, 277
 - WeakConnectivityDFS, 284
- Exists
 - Objects, 142
- Explode
 - Graph, 105
- Export
 - Graph, 109
- ExportManager, 84
 - EnableType, 87
 - GetEdge, 87
 - GetEdgeType, 86
 - GetGraph, 86
 - GetNode, 86
 - GetNodeType, 86
 - Prepare, 85
 - Release, 85
- ExportType
 - gdb, 17
- FileNotFoundException, 88
 - FileNotFoundException, 89
 - Message, 89
 - SetMessage, 89
- FindAttribute
 - Graph, 102
- FindAttributes
 - Graph, 108
- FindEdge
 - Graph, 107
- FindEdgeTypes
 - Graph, 108
- FindNodeTypes
 - Graph, 108
- FindObject
 - Graph, 103
- FindOrCreateEdge
 - Graph, 107
- FindOrCreateObject
 - Graph, 103
- FindType
 - Graph, 101
- FindTypes
 - Graph, 108
- Fine
 - gdb, 18
- Gdb, 7
 - gdb
 - ~Handler, 20
 - Any, 17
 - Ascendent, 19
 - AttributeKind, 15
 - Basic, 15
 - Between, 16
 - Boolean, 17
 - Box, 18
 - ColorRGB, 15
 - Condition, 15
 - Config, 18
 - DataType, 16
 - Debug, 18
 - Descendent, 19
 - Double, 17
 - Edge, 18
 - EdgesDirection, 17
 - Equal, 16
 - ExportType, 17
 - Fine, 18
 - GetHandler, 19
 - GraphML, 17
 - Graphviz, 17
 - GreaterEqual, 16
 - GreaterThan, 16
 - Handler, 19
 - Indexed, 15
 - Info, 18
 - Ingoing, 17
 - Integer, 17
 - IsNull, 19
 - LessEqual, 16

- LessThan, 16
- Like, 16
- LikeNoCase, 16
- LogLevel, 17
- Long, 17
- Node, 18
- NodeShape, 18
- NotEqual, 16
- ObjectType, 18
- Off, 18
- OID, 17
- operator<<, 19
- Order, 18
- Outgoing, 17
- RegExp, 16
- Round, 18
- SetHandler, 20
- Severe, 18
- String, 17
- Text, 17
- Timestamp, 17
- Unique, 15
- Warning, 18
- YGraphML, 17
- GenerateSchemaScript
 - ScriptParser, 159
- Get
 - SparkseeProperties, 199
- GetAlias
 - Database, 59
- GetAreNeighborsIndexed
 - Type, 236
- GetAttribute
 - Graph, 99, 102
- GetAttributeIntervalCount
 - Graph, 101
- GetAttributes
 - Graph, 108
- GetAttributeStatistics
 - Graph, 100
- GetAttributeText
 - Graph, 100
- GetAvailableMem
 - PlatformStatistics, 153
- GetAvgLengthString
 - AttributeStatistics, 35
- GetBoolean
 - SparkseeProperties, 200
 - Value, 264
- GetCache
 - DatabaseStatistics, 61
- GetCacheMaxSize
 - SparkseeConfig, 193
- GetCacheStatisticsEnabled
 - SparkseeConfig, 194
- GetCacheStatisticsFile
 - SparkseeConfig, 194
- GetCacheStatisticsSnapshotTime
 - SparkseeConfig, 195
- GetColorRGB
 - EdgeExport, 68
 - NodeExport, 124
- GetConnectedComponent
 - ConnectedComponents, 40
- GetConnectedComponents
 - Connectivity, 45
 - StrongConnectivity, 209
 - StrongConnectivityGabow, 215
 - WeakConnectivity, 278
 - WeakConnectivityDFS, 284
- GetCost
 - SinglePairShortestPath, 170
 - SinglePairShortestPathBFS, 176
 - SinglePairShortestPathDijkstra, 182
- GetCount
 - Attribute, 29
 - ConnectedComponents, 40
 - TypeExporterEvent, 243
 - TypeLoaderEvent, 255
- GetCurrentDepth
 - Traversal, 225
 - TraversalBFS, 228
 - TraversalDFS, 233
- GetData
 - DatabaseStatistics, 61
- GetDataType
 - Attribute, 29
 - Value, 263
- GetDistinct
 - AttributeStatistics, 34
- GetDouble
 - Value, 264
- GetEdge
 - DefaultExport, 64
 - EdgeData, 65
 - ExportManager, 87
- GetEdgeData
 - Graph, 96
- GetEdgePeer
 - Graph, 97
- GetEdgeType
 - DefaultExport, 63
 - ExportManager, 86
- GetExtentPages
 - SparkseeConfig, 191
- GetExtentSize
 - SparkseeConfig, 191
- GetFontSize

- EdgeExport, 69
- NodeExport, 126
- GetGraph
 - DefaultExport, 63
 - ExportManager, 86
 - Session, 162
- GetHandler
 - gdb, 19
- GetHead
 - EdgeData, 65
- GetHeight
 - NodeExport, 125
- GetHighAvailabilityCoordinators
 - SparkseeConfig, 197
- GetHighAvailabilityEnabled
 - SparkseeConfig, 196
- GetHighAvailabilityIP
 - SparkseeConfig, 197
- GetHighAvailabilityMasterHistory
 - SparkseeConfig, 198
- GetHighAvailabilitySynchronization
 - SparkseeConfig, 197
- GetId
 - Attribute, 28
 - Type, 236
- GetInteger
 - SparkseeProperties, 199
 - Value, 264
- GetIsDirected
 - Type, 236
- GetIsRestricted
 - Type, 236
- GetKind
 - Attribute, 29
- GetLabel
 - EdgeExport, 67
 - GraphExport, 111
 - NodeExport, 124
- GetLabelColorRGB
 - EdgeExport, 68
 - NodeExport, 124
- GetLicense
 - SparkseeConfig, 193
- GetLogFile
 - SparkseeConfig, 193
- GetLogLevel
 - SparkseeConfig, 194
- GetLong
 - Value, 264
- GetMax
 - AttributeStatistics, 34
- GetMaxLengthString
 - AttributeStatistics, 34
- GetMean
 - AttributeStatistics, 35
- GetMedian
 - AttributeStatistics, 36
- GetMin
 - AttributeStatistics, 34
- GetMinLengthString
 - AttributeStatistics, 35
- GetMode
 - AttributeStatistics, 35
- GetModeCount
 - AttributeStatistics, 35
- GetName
 - Attribute, 28
 - Type, 236
- GetNode
 - DefaultExport, 64
 - ExportManager, 86
- GetNodes
 - ConnectedComponents, 40
- GetNodeType
 - DefaultExport, 63
 - ExportManager, 86
- GetNull
 - AttributeStatistics, 34
- GetNumCPUs
 - PlatformStatistics, 153
- GetNumObjects
 - Type, 236
- GetObjectType
 - Graph, 97
 - Type, 236
- GetOID
 - Value, 265
- GetPartition
 - TypeLoaderEvent, 255
- GetPath
 - Database, 59
- GetPathAsEdges
 - SinglePairShortestPath, 170
 - SinglePairShortestPathBFS, 175
 - SinglePairShortestPathDijkstra, 181
- GetPathAsNodes
 - SinglePairShortestPath, 170
 - SinglePairShortestPathBFS, 175
 - SinglePairShortestPathDijkstra, 181
- GetPhase
 - TypeLoaderEvent, 255
- GetPoolFrameSize
 - SparkseeConfig, 191
- GetPoolPersistentMaxSize
 - SparkseeConfig, 192
- GetPoolPersistentMinSize
 - SparkseeConfig, 192
- GetPoolTemporaryMaxSize

- SparkseeConfig, 193
- GetPoolTemporaryMinSize
 - SparkseeConfig, 192
- GetRead
 - DatabaseStatistics, 61
- GetRealTime
 - PlatformStatistics, 153
- GetRecoveryCacheMaxSize
 - SparkseeConfig, 196
- GetRecoveryCheckpointTime
 - SparkseeConfig, 196
- GetRecoveryEnabled
 - SparkseeConfig, 195
- GetRecoveryLogFile
 - SparkseeConfig, 195
- GetRestrictedFrom
 - Type, 237
- GetRestrictedTo
 - Type, 237
- GetRow
 - CSVReader, 54
 - RowReader, 155
- GetSessions
 - DatabaseStatistics, 61
- GetShape
 - NodeExport, 124
- GetSize
 - Attribute, 29
 - ConnectedComponents, 40
- GetStatistics
 - Database, 59
 - Platform, 152
- GetString
 - Value, 264
- GetSystemTime
 - PlatformStatistics, 153
- GetTail
 - EdgeData, 65
- GetTemp
 - DatabaseStatistics, 61
- GetTimestamp
 - Value, 264
- GetTimeUnit
 - SparkseeProperties, 200
- GetTotal
 - AttributeStatistics, 34
 - TypeExporterEvent, 243
- GetTotalMem
 - PlatformStatistics, 153
- GetTotalPartitions
 - TypeLoaderEvent, 255
- GetTotalPartitionSteps
 - TypeLoaderEvent, 256
- GetTotalPhases
 - TypeLoaderEvent, 255
- GetType
 - Graph, 101
- GetTypeId
 - Attribute, 28
 - TypeExporterEvent, 243
 - TypeLoaderEvent, 255
- GetUserTime
 - PlatformStatistics, 153
- GetValues
 - Graph, 109
- GetVariance
 - AttributeStatistics, 35
- GetWidth
 - EdgeExport, 68
 - NodeExport, 125
- GetWrite
 - DatabaseStatistics, 61
- Graph, 89
 - Backup, 109
 - CountEdges, 96
 - CountNodes, 96
 - Degree, 106
 - Drop, 97
 - DumpData, 109
 - DumpStorage, 109
 - Edges, 106
 - Explode, 105
 - Export, 109
 - FindAttribute, 102
 - FindAttributes, 108
 - FindEdge, 107
 - FindEdgeTypes, 108
 - FindNodeTypes, 108
 - FindObject, 103
 - FindOrCreateEdge, 107
 - FindOrCreateObject, 103
 - FindType, 101
 - FindTypes, 108
 - GetAttribute, 99, 102
 - GetAttributeIntervalCount, 101
 - GetAttributes, 108
 - GetAttributeStatistics, 100
 - GetAttributeText, 100
 - GetEdgeData, 96
 - GetEdgePeer, 97
 - GetObjectType, 97
 - GetType, 101
 - GetValues, 109
 - Heads, 107
 - IndexAttribute, 99
 - Neighbors, 106
 - NewAttribute, 97, 98
 - NewEdge, 95, 96

- NewEdgeType, 95
- NewNode, 95
- NewNodeType, 95
- NewRestrictedEdgeType, 95
- NewSessionAttribute, 98
- RemoveAttribute, 103
- RemoveType, 102
- RenameAttribute, 103
- RenameType, 102
- Select, 104, 105
- SetAttribute, 100
- SetAttributeDefaultValue, 99
- SetAttributeText, 100
- Tails, 107
- TailsAndHeads, 108
- GraphExport, 110
 - GetLabel, 111
 - SetLabel, 111
- GraphML
 - gdb, 17
- Graphviz
 - gdb, 17
- GreaterEqual
 - gdb, 16
- GreaterThan
 - gdb, 16
- Handler, 111
 - gdb, 19
- HasNext
 - AttributeListIterator, 32
 - BooleanListIterator, 38
 - Int32ListIterator, 115
 - ObjectsIterator, 148
 - OIDListIterator, 151
 - StringListIterator, 205
 - Traversal, 225
 - TraversalBFS, 228
 - TraversalDFS, 233
 - TypeListIterator, 247
 - ValuesIterator, 273
- Heads
 - Graph, 107
- IndexAttribute
 - Graph, 99
- Indexed
 - gdb, 15
- Info
 - gdb, 18
- Ingoing
 - gdb, 17
- Int32List, 113
 - Add, 114
 - Count, 114
 - Int32List, 113
 - Iterator, 114
- Int32ListIterator, 114
 - HasNext, 115
 - Next, 115
- Integer
 - gdb, 17
- Intersection
 - Objects, 143
- Io, 20
- IOError, 115
 - IOError, 117
 - Message, 117
 - NewError, 117
 - SetMessage, 117
- IOException, 118
 - IOException, 119
 - Message, 119
 - SetMessage, 119
- IsFit
 - NodeExport, 125
- IsLast
 - TypeExporterEvent, 243
 - TypeLoaderEvent, 256
- IsNull
 - gdb, 19
 - TextStream, 221
 - Value, 263
- IsSessionAttribute
 - Attribute, 29
- Iterator
 - AttributeList, 31
 - BooleanList, 37
 - Int32List, 114
 - Objects, 145
 - OIDList, 149
 - StringList, 203
 - TypeList, 246
 - Values, 271
- IteratorFromElement
 - Objects, 146
- IteratorFromIndex
 - Objects, 145
- LessEqual
 - gdb, 16
- LessThan
 - gdb, 16
- LicenseError, 119
 - LicenseError, 121
 - Message, 121
 - NewError, 121
 - SetMessage, 121

- Like
 - [gdb, 16](#)
- LikeNoCase
 - [gdb, 16](#)
- Load
 - [SparkseeProperties, 199](#)
- LogLevel
 - [gdb, 17](#)
- Long
 - [gdb, 17](#)
- maxHops
 - [ShortestPath, 166](#)
 - [SinglePairShortestPath, 172](#)
 - [SinglePairShortestPathBFS, 177](#)
 - [SinglePairShortestPathDijkstra, 183](#)
- Message
 - [AppError, 27](#)
 - [Error, 82](#)
 - [Exception, 84](#)
 - [FileNotFoundException, 89](#)
 - [IOError, 117](#)
 - [IOException, 119](#)
 - [LicenseError, 121](#)
 - [NoSuchElementException, 138](#)
 - [SQLException, 202](#)
 - [SystemError, 218](#)
 - [UnsupportedOperationException, 259](#)
 - [WrongArgumentError, 287](#)
- Mode
 - [EdgeTypeLoader, 76](#)
 - [NodeTypeLoader, 133](#)
 - [TypeLoader, 250](#)
- N_PHASES
 - [EdgeTypeLoader, 76](#)
 - [NodeTypeLoader, 133](#)
 - [TypeLoader, 250](#)
- Neighbors
 - [Graph, 106](#)
- NewAttribute
 - [Graph, 97, 98](#)
- NewEdge
 - [Graph, 95, 96](#)
- NewEdgeType
 - [Graph, 95](#)
- NewError
 - [AppError, 26](#)
 - [Error, 82](#)
 - [IOError, 117](#)
 - [LicenseError, 121](#)
 - [SystemError, 218](#)
 - [UnsupportedOperationException, 259](#)
 - [WrongArgumentError, 287](#)
- NewNode
 - [Graph, 95](#)
- NewNodeType
 - [Graph, 95](#)
- NewObjects
 - [Session, 162](#)
- NewRestrictedEdgeType
 - [Graph, 95](#)
- NewSessionAttribute
 - [Graph, 98](#)
- Next
 - [AttributeListIterator, 32](#)
 - [BooleanListIterator, 38](#)
 - [Int32ListIterator, 115](#)
 - [ObjectsIterator, 148](#)
 - [OIDListIterator, 151](#)
 - [StringListIterator, 205](#)
 - [Traversal, 224](#)
 - [TraversalBFS, 228](#)
 - [TraversalDFS, 233](#)
 - [TypeListIterator, 247](#)
 - [ValuesIterator, 273](#)
- Node
 - [gdb, 18](#)
- NodeExport, 122
 - [GetColorRGB, 124](#)
 - [GetFontSize, 126](#)
 - [GetHeight, 125](#)
 - [GetLabel, 124](#)
 - [GetLabelColorRGB, 124](#)
 - [GetShape, 124](#)
 - [GetWidth, 125](#)
 - [IsFit, 125](#)
 - [SetColorRGB, 124](#)
 - [SetFit, 125](#)
 - [SetFontSize, 126](#)
 - [SetHeight, 125](#)
 - [SetLabel, 124](#)
 - [SetLabelColorRGB, 125](#)
 - [SetShape, 124](#)
 - [SetWidth, 125](#)
- NodeShape
 - [gdb, 18](#)
- NodeTypeExporter, 126
 - [CanRun, 128](#)
 - [NodeTypeExporter, 128](#)
 - [NotifyListeners, 128](#)
 - [Register, 129](#)
 - [RunProcess, 128](#)
 - [SetAttributes, 130](#)
 - [SetFrequency, 130](#)
 - [SetGraph, 130](#)
 - [SetHeadAttribute, 129](#)
 - [SetHeader, 130](#)

- SetHeadPosition, 129
- SetRowWriter, 130
- SetTailAttribute, 129
- SetTailPosition, 129
- SetType, 130
- NodeTypeLoader, 131
 - CanRun, 134
 - Mode, 133
 - N_PHASES, 133
 - NodeTypeLoader, 134
 - NotifyListeners, 134
 - ONE_PHASE, 133
 - Register, 136
 - Run, 134
 - SetAttributePositions, 137
 - SetAttributes, 136
 - SetFrequency, 137
 - SetGraph, 136
 - SetHeadAttribute, 134
 - SetHeadPosition, 135
 - SetLocale, 136
 - SetLogError, 135
 - SetLogOff, 136
 - SetRowReader, 136
 - SetTailAttribute, 135
 - SetTailPosition, 135
 - SetType, 136
 - TWO_PHASES, 133
- NoSuchElementException, 137
 - Message, 138
 - NoSuchElementException, 138
 - SetMessage, 138
- NotEqual
 - gdb, 16
- NotifyEvent
 - TypeExporterListener, 244
 - TypeLoaderListener, 257
- NotifyListeners
 - EdgeTypeExporter, 72
 - EdgeTypeLoader, 78
 - NodeTypeExporter, 128
 - NodeTypeLoader, 134
 - TypeExporter, 239
 - TypeLoader, 251
- Objects, 139
 - Add, 142
 - Any, 142
 - CombineDifference, 144
 - CombineIntersection, 144
 - CombineUnion, 144
 - Contains, 144
 - Copy, 141, 145
 - Count, 142
 - Difference, 143
 - Equals, 143
 - Exists, 142
 - Intersection, 143
 - Iterator, 145
 - IteratorFromElement, 146
 - IteratorFromIndex, 145
 - Remove, 142
 - Sample, 145
 - Union, 143
- ObjectsIterator, 146
 - HasNext, 148
 - Next, 148
- ObjectType
 - gdb, 18
- Off
 - gdb, 18
- OID
 - gdb, 17
- OIDList, 148
 - Add, 150
 - Count, 149
 - Iterator, 149
 - OIDList, 149
 - Set, 150
- OIDListIterator, 150
 - HasNext, 151
 - Next, 151
- ONE_PHASE
 - EdgeTypeLoader, 76
 - NodeTypeLoader, 133
 - TypeLoader, 250
- Open
 - CSVReader, 53
 - CSVWriter, 57
 - Sparksee, 185
- operator<<
 - gdb, 19
 - script, 23
- operator=
 - Value, 263
- Order
 - gdb, 18
- Outgoing
 - gdb, 17
- Parse
 - ScriptParser, 159
- pathAsEdges
 - SinglePairShortestPath, 172
 - SinglePairShortestPathBFS, 177
 - SinglePairShortestPathDijkstra, 183
- Platform, 151
 - GetStatistics, 152

- PlatformStatistics, 152
 - GetAvailableMem, 153
 - GetNumCPUs, 153
 - GetRealTime, 153
 - GetSystemTime, 153
 - GetTotalMem, 153
 - GetUserTime, 153
- Prepare
 - ExportManager, 85
- Read
 - CSVReader, 53
 - RowReader, 155
 - TextStream, 220
- RegExp
 - gdb, 16
- Register
 - EdgeTypeExporter, 72
 - EdgeTypeLoader, 79
 - NodeTypeExporter, 129
 - NodeTypeLoader, 136
 - TypeExporter, 241
 - TypeLoader, 252
- Release
 - ExportManager, 85
- Remove
 - Objects, 142
- RemoveAttribute
 - Graph, 103
- RemoveType
 - Graph, 102
- RenameAttribute
 - Graph, 103
- RenameType
 - Graph, 102
- Reset
 - CSVReader, 53
 - RowReader, 155
- Restore
 - Sparksee, 186
- Round
 - gdb, 18
- RowReader, 154
 - Close, 155
 - GetRow, 155
 - Read, 155
 - Reset, 155
 - RowReader, 155
- RowWriter, 156
 - Close, 157
 - RowWriter, 157
 - Write, 157
- Run
 - Connectivity, 45
 - EdgeTypeLoader, 78
 - NodeTypeLoader, 134
 - ShortestPath, 166
 - SinglePairShortestPath, 171
 - StrongConnectivity, 210
 - TypeExporter, 241
 - TypeLoader, 250
 - WeakConnectivity, 278
- RunNPhases
 - TypeLoader, 252
- RunProcess
 - EdgeTypeExporter, 72
 - NodeTypeExporter, 128
 - TypeExporter, 240
- RunTwoPhases
 - TypeLoader, 252
- Sample
 - Objects, 145
- Script, 22
- script
 - operator<<, 23
- ScriptParser, 157
 - GenerateSchemaScript, 159
 - Parse, 159
 - SetErrorLog, 159
 - SetOutputLog, 158
- Select
 - Graph, 104, 105
- Session, 160
 - GetGraph, 162
 - NewObjects, 162
- Set
 - OIDList, 150
 - Value, 268
- SetAsDirected
 - EdgeExport, 67
- SetAttribute
 - Graph, 100
- SetAttributeDefaultValue
 - Graph, 99
- SetAttributePositions
 - EdgeTypeLoader, 80
 - NodeTypeLoader, 137
 - TypeLoader, 253
- SetAttributes
 - EdgeTypeExporter, 73
 - EdgeTypeLoader, 80
 - NodeTypeExporter, 130
 - NodeTypeLoader, 136
 - TypeExporter, 241
 - TypeLoader, 253
- SetAttributeText
 - Graph, 100

- SetAutoQuotes
 - CSVWriter, 56
- SetBoolean
 - Value, 265
- SetBooleanVoid
 - Value, 265
- SetCacheMaxSize
 - SparkseeConfig, 193
- SetCacheStatisticsEnabled
 - SparkseeConfig, 194
- SetCacheStatisticsFile
 - SparkseeConfig, 195
- SetCacheStatisticsSnapshotTime
 - SparkseeConfig, 195
- SetColorRGB
 - EdgeExport, 68
 - NodeExport, 124
- SetDouble
 - Value, 266
- SetDoubleVoid
 - Value, 266
- SetErrorLog
 - ScriptParser, 159
- SetExtentPages
 - SparkseeConfig, 191
- SetExtentSize
 - SparkseeConfig, 191
- SetFit
 - NodeExport, 125
- SetFontSize
 - EdgeExport, 69
 - NodeExport, 126
- SetForcedQuotes
 - CSVWriter, 56
- SetFrequency
 - EdgeTypeExporter, 73
 - EdgeTypeLoader, 80
 - NodeTypeExporter, 130
 - NodeTypeLoader, 137
 - TypeExporter, 242
 - TypeLoader, 253
- SetGraph
 - EdgeTypeExporter, 73
 - EdgeTypeLoader, 79
 - NodeTypeExporter, 130
 - NodeTypeLoader, 136
 - TypeExporter, 241
 - TypeLoader, 253
- SetHandler
 - gdb, 20
- SetHeadAttribute
 - EdgeTypeExporter, 71
 - EdgeTypeLoader, 77
 - NodeTypeExporter, 129
 - NodeTypeLoader, 134
 - TypeExporter, 240
 - TypeLoader, 251
- SetHeader
 - EdgeTypeExporter, 73
 - NodeTypeExporter, 130
 - TypeExporter, 242
- SetHeadPosition
 - EdgeTypeExporter, 71
 - EdgeTypeLoader, 77
 - NodeTypeExporter, 129
 - NodeTypeLoader, 135
 - TypeExporter, 240
 - TypeLoader, 251
- SetHeight
 - NodeExport, 125
- SetHighAvailabilityCoordinators
 - SparkseeConfig, 197
- SetHighAvailabilityEnabled
 - SparkseeConfig, 197
- SetHighAvailabilityIP
 - SparkseeConfig, 197
- SetHighAvailabilityMasterHistory
 - SparkseeConfig, 198
- SetHighAvailabilitySynchronization
 - SparkseeConfig, 198
- SetInteger
 - Value, 265
- SetIntegerVoid
 - Value, 265
- SetLabel
 - EdgeExport, 67
 - GraphExport, 111
 - NodeExport, 124
- SetLabelColorRGB
 - EdgeExport, 68
 - NodeExport, 125
- SetLicense
 - SparkseeConfig, 193
- SetLocale
 - CSVReader, 53
 - CSVWriter, 56
 - EdgeTypeLoader, 79
 - NodeTypeLoader, 136
 - TypeLoader, 253
- SetLogError
 - EdgeTypeLoader, 78
 - NodeTypeLoader, 135
 - TypeLoader, 252
- SetLogFile
 - SparkseeConfig, 194
- SetLogLevel
 - SparkseeConfig, 194
- SetLogOff

- EdgeTypeLoader, [79](#)
- NodeTypeLoader, [136](#)
- TypeLoader, [252](#)
- SetLong
 - Value, [266](#)
- SetLongVoid
 - Value, [265](#)
- SetMaterializedAttribute
 - Connectivity, [45](#)
 - StrongConnectivity, [210](#)
 - StrongConnectivityGabow, [215](#)
 - WeakConnectivity, [278](#)
 - WeakConnectivityDFS, [285](#)
- SetMaximumHops
 - Context, [49](#)
 - ShortestPath, [165](#)
 - SinglePairShortestPath, [170](#)
 - SinglePairShortestPathBFS, [176](#)
 - SinglePairShortestPathDijkstra, [182](#)
 - Traversal, [225](#)
 - TraversalBFS, [229](#)
 - TraversalDFS, [234](#)
- SetMessage
 - AppError, [27](#)
 - Error, [82](#)
 - Exception, [84](#)
 - FileNotFoundException, [89](#)
 - IOError, [117](#)
 - IOException, [119](#)
 - LicenseError, [121](#)
 - NoSuchElementException, [138](#)
 - SQLException, [202](#)
 - SystemError, [218](#)
 - UnsupportedOperationError, [259](#)
 - WrongArgumentError, [287](#)
- SetMultilines
 - CSVReader, [52](#)
- SetNodesNotVisited
 - Connectivity, [46](#)
 - StrongConnectivity, [210](#)
 - StrongConnectivityGabow, [216](#)
 - WeakConnectivity, [279](#)
 - WeakConnectivityDFS, [285](#)
- SetNull
 - Value, [263](#)
- SetNumLines
 - CSVReader, [53](#)
- SetOID
 - Value, [268](#)
- SetOIDVoid
 - Value, [268](#)
- SetOutputLog
 - ScriptParser, [158](#)
- SetPoolFrameSize
 - SparkseeConfig, [191](#)
- SetPoolPersistentMaxSize
 - SparkseeConfig, [192](#)
- SetPoolPersistentMinSize
 - SparkseeConfig, [192](#)
- SetPoolTemporaryMaxSize
 - SparkseeConfig, [193](#)
- SetPoolTemporaryMinSize
 - SparkseeConfig, [192](#)
- SetQuotes
 - CSVReader, [52](#)
 - CSVWriter, [56](#)
- SetRecoveryCacheMaxSize
 - SparkseeConfig, [196](#)
- SetRecoveryCheckpointTime
 - SparkseeConfig, [196](#)
- SetRecoveryEnabled
 - SparkseeConfig, [195](#)
- SetRecoveryLogFile
 - SparkseeConfig, [196](#)
- SetRowReader
 - EdgeTypeLoader, [79](#)
 - NodeTypeLoader, [136](#)
 - TypeLoader, [253](#)
- SetRowWriter
 - EdgeTypeExporter, [73](#)
 - NodeTypeExporter, [130](#)
 - TypeExporter, [241](#)
- SetSeparator
 - CSVReader, [52](#)
 - CSVWriter, [56](#)
- SetShape
 - NodeExport, [124](#)
- SetStartLine
 - CSVReader, [52](#)
- SetString
 - Value, [267](#)
- SetStringVoid
 - Value, [267](#)
- SetTailAttribute
 - EdgeTypeExporter, [72](#)
 - EdgeTypeLoader, [77](#)
 - NodeTypeExporter, [129](#)
 - NodeTypeLoader, [135](#)
 - TypeExporter, [240](#)
 - TypeLoader, [251](#)
- SetTailPosition
 - EdgeTypeExporter, [72](#)
 - EdgeTypeLoader, [78](#)
 - NodeTypeExporter, [129](#)
 - NodeTypeLoader, [135](#)
 - TypeExporter, [240](#)
 - TypeLoader, [251](#)
- SetTimestamp

- Value, [267](#)
- SetTimestampVoid
 - Value, [266](#)
- SetType
 - EdgeTypeExporter, [73](#)
 - EdgeTypeLoader, [79](#)
 - NodeTypeExporter, [130](#)
 - NodeTypeLoader, [136](#)
 - TypeExporter, [241](#)
 - TypeLoader, [253](#)
- SetUnweightedEdgeCost
 - SinglePairShortestPathDijkstra, [182](#)
- SetVoid
 - Value, [268](#)
- SetWidth
 - EdgeExport, [68](#)
 - NodeExport, [125](#)
- Severe
 - gdb, [18](#)
- ShortestPath, [162](#)
 - AddAllEdgeTypes, [165](#)
 - AddEdgeType, [165](#)
 - ExcludeEdges, [166](#)
 - ExcludeNodes, [166](#)
 - maxHops, [166](#)
 - Run, [166](#)
 - SetMaximumHops, [165](#)
 - ShortestPath, [165](#)
- SinglePairShortestPath, [167](#)
 - AddAllEdgeTypes, [171](#)
 - AddEdgeType, [171](#)
 - ExcludeEdges, [171](#)
 - ExcludeNodes, [171](#)
 - GetCost, [170](#)
 - GetPathAsEdges, [170](#)
 - GetPathAsNodes, [170](#)
 - maxHops, [172](#)
 - pathAsEdges, [172](#)
 - Run, [171](#)
 - SetMaximumHops, [170](#)
 - SinglePairShortestPath, [170](#)
- SinglePairShortestPathBFS, [172](#)
 - AddAllEdgeTypes, [176](#)
 - AddEdgeType, [176](#)
 - CheckOnlyExistence, [176](#)
 - ExcludeEdges, [177](#)
 - ExcludeNodes, [177](#)
 - GetCost, [176](#)
 - GetPathAsEdges, [175](#)
 - GetPathAsNodes, [175](#)
 - maxHops, [177](#)
 - pathAsEdges, [177](#)
 - SetMaximumHops, [176](#)
 - SinglePairShortestPathBFS, [175](#)
- SinglePairShortestPathDijkstra, [177](#)
 - AddAllEdgeTypes, [183](#)
 - AddEdgeType, [182](#)
 - AddWeightedEdgeType, [182](#)
 - ExcludeEdges, [183](#)
 - ExcludeNodes, [183](#)
 - GetCost, [182](#)
 - GetPathAsEdges, [181](#)
 - GetPathAsNodes, [181](#)
 - maxHops, [183](#)
 - pathAsEdges, [183](#)
 - SetMaximumHops, [182](#)
 - SetUnweightedEdgeCost, [182](#)
 - SinglePairShortestPathDijkstra, [181](#)
- Sparksee, [184](#)
 - Create, [185](#)
 - Open, [185](#)
 - Restore, [186](#)
 - Sparksee, [185](#)
- SparkseeConfig, [186](#)
 - GetCacheMaxSize, [193](#)
 - GetCacheStatisticsEnabled, [194](#)
 - GetCacheStatisticsFile, [194](#)
 - GetCacheStatisticsSnapshotTime, [195](#)
 - GetExtentPages, [191](#)
 - GetExtentSize, [191](#)
 - GetHighAvailabilityCoordinators, [197](#)
 - GetHighAvailabilityEnabled, [196](#)
 - GetHighAvailabilityIP, [197](#)
 - GetHighAvailabilityMasterHistory, [198](#)
 - GetHighAvailabilitySynchronization, [197](#)
 - GetLicense, [193](#)
 - GetLogFile, [193](#)
 - GetLogLevel, [194](#)
 - GetPoolFrameSize, [191](#)
 - GetPoolPersistentMaxSize, [192](#)
 - GetPoolPersistentMinSize, [192](#)
 - GetPoolTemporaryMaxSize, [193](#)
 - GetPoolTemporaryMinSize, [192](#)
 - GetRecoveryCacheMaxSize, [196](#)
 - GetRecoveryCheckpointTime, [196](#)
 - GetRecoveryEnabled, [195](#)
 - GetRecoveryLogFile, [195](#)
 - SetCacheMaxSize, [193](#)
 - SetCacheStatisticsEnabled, [194](#)
 - SetCacheStatisticsFile, [195](#)
 - SetCacheStatisticsSnapshotTime, [195](#)
 - SetExtentPages, [191](#)
 - SetExtentSize, [191](#)
 - SetHighAvailabilityCoordinators, [197](#)
 - SetHighAvailabilityEnabled, [197](#)
 - SetHighAvailabilityIP, [197](#)
 - SetHighAvailabilityMasterHistory, [198](#)
 - SetHighAvailabilitySynchronization, [198](#)

- SetLicense, 193
- SetLogFile, 194
- SetLogLevel, 194
- SetPoolFrameSize, 191
- SetPoolPersistentMaxSize, 192
- SetPoolPersistentMinSize, 192
- SetPoolTemporaryMaxSize, 193
- SetPoolTemporaryMinSize, 192
- SetRecoveryCacheMaxSize, 196
- SetRecoveryCheckpointTime, 196
- SetRecoveryEnabled, 195
- SetRecoveryLogFile, 196
- SparkseeConfig, 191
- SparkseeProperties, 198
 - Get, 199
 - GetBoolean, 200
 - GetInteger, 199
 - GetTimeUnit, 200
 - Load, 199
- SQLException, 200
 - Message, 202
 - SetMessage, 202
 - SQLException, 202
- String
 - gdb, 17
- StringList, 202
 - Add, 204
 - Count, 203
 - Iterator, 203
 - StringList, 203
- StringListIterator, 204
 - HasNext, 205
 - Next, 205
- StrongConnectivity, 205
 - AddAllEdgeTypes, 209
 - AddEdgeType, 209
 - ExcludeEdges, 209
 - ExcludeNodes, 209
 - GetConnectedComponents, 209
 - Run, 210
 - SetMaterializedAttribute, 210
 - SetNodesNotVisited, 210
 - StrongConnectivity, 209
- StrongConnectivityGabow, 210
 - AddAllEdgeTypes, 215
 - AddEdgeType, 214
 - ExcludeEdges, 215
 - ExcludeNodes, 215
 - GetConnectedComponents, 215
 - SetMaterializedAttribute, 215
 - SetNodesNotVisited, 216
 - StrongConnectivityGabow, 214
- SystemError, 216
 - Message, 218
 - NewError, 218
 - SetMessage, 218
 - SystemError, 218
- Tails
 - Graph, 107
- TailsAndHeads
 - Graph, 108
- Text
 - gdb, 17
- TextStream, 218
 - Close, 221
 - IsNull, 221
 - Read, 220
 - TextStream, 220
 - Write, 221
- Timestamp
 - gdb, 17
- ToString
 - Value, 269
- Traversal, 221
 - AddAllEdgeTypes, 224
 - AddEdgeType, 224
 - ExcludeEdges, 224
 - ExcludeNodes, 224
 - GetCurrentDepth, 225
 - HasNext, 225
 - Next, 224
 - SetMaximumHops, 225
 - Traversal, 224
- TraversalBFS, 225
 - AddAllEdgeTypes, 229
 - AddEdgeType, 229
 - ExcludeEdges, 229
 - ExcludeNodes, 229
 - GetCurrentDepth, 228
 - HasNext, 228
 - Next, 228
 - SetMaximumHops, 229
 - TraversalBFS, 228
- TraversalDFS, 230
 - AddAllEdgeTypes, 233
 - AddEdgeType, 233
 - ExcludeEdges, 234
 - ExcludeNodes, 233
 - GetCurrentDepth, 233
 - HasNext, 233
 - Next, 233
 - SetMaximumHops, 234
 - TraversalDFS, 232
- TWO_PHASES
 - EdgeTypeLoader, 76
 - NodeTypeLoader, 133
 - TypeLoader, 250

- Type, 234
 - GetAreNeighborsIndexed, 236
 - GetId, 236
 - GetIsDirected, 236
 - GetIsRestricted, 236
 - GetName, 236
 - GetNumObjects, 236
 - GetObjectType, 236
 - GetRestrictedFrom, 237
 - GetRestrictedTo, 237
- TypeExporter, 237
 - CanRun, 239
 - NotifyListeners, 239
 - Register, 241
 - Run, 241
 - RunProcess, 240
 - SetAttributes, 241
 - SetFrequency, 242
 - SetGraph, 241
 - SetHeadAttribute, 240
 - SetHeader, 242
 - SetHeadPosition, 240
 - SetRowWriter, 241
 - SetTailAttribute, 240
 - SetTailPosition, 240
 - SetType, 241
 - TypeExporter, 239
- TypeExporterEvent, 242
 - GetCount, 243
 - GetTotal, 243
 - GetTypeId, 243
 - IsLast, 243
- TypeExporterListener, 243
 - NotifyEvent, 244
 - TypeExporterListener, 244
- TypeList, 244
 - Add, 246
 - Count, 246
 - Iterator, 246
 - TypeList, 245
- TypeListIterator, 246
 - HasNext, 247
 - Next, 247
- TypeLoader, 247
 - CanRun, 250
 - Mode, 250
 - N_PHASES, 250
 - NotifyListeners, 251
 - ONE_PHASE, 250
 - Register, 252
 - Run, 250
 - RunNPhases, 252
 - RunTwoPhases, 252
 - SetAttributePositions, 253
 - SetAttributes, 253
 - SetFrequency, 253
 - SetGraph, 253
 - SetHeadAttribute, 251
 - SetHeadPosition, 251
 - SetLocale, 253
 - SetLogError, 252
 - SetLogOff, 252
 - SetRowReader, 253
 - SetTailAttribute, 251
 - SetTailPosition, 251
 - SetType, 253
 - TWO_PHASES, 250
 - TypeLoader, 250
- TypeLoaderEvent, 254
 - GetCount, 255
 - GetPartition, 255
 - GetPhase, 255
 - GetTotalPartitions, 255
 - GetTotalPartitionSteps, 256
 - GetTotalPhases, 255
 - GetTypeId, 255
 - IsLast, 256
- TypeLoaderListener, 256
 - NotifyEvent, 257
 - TypeLoaderListener, 257
- Union
 - Objects, 143
- Unique
 - gdb, 15
- UnsupportedOperationException, 257
 - Message, 259
 - NewError, 259
 - SetMessage, 259
 - UnsupportedOperationException, 259
- Value, 259
 - Compare, 268
 - Equals, 269
 - GetBoolean, 264
 - GetDataType, 263
 - GetDouble, 264
 - GetInteger, 264
 - GetLong, 264
 - GetOID, 265
 - GetString, 264
 - GetTimestamp, 264
 - IsNull, 263
 - operator=, 263
 - Set, 268
 - SetBoolean, 265
 - SetBooleanVoid, 265
 - SetDouble, 266

- SetDoubleVoid, [266](#)
- SetInteger, [265](#)
- SetIntegerVoid, [265](#)
- SetLong, [266](#)
- SetLongVoid, [265](#)
- SetNull, [263](#)
- SetOID, [268](#)
- SetOIDVoid, [268](#)
- SetString, [267](#)
- SetStringVoid, [267](#)
- SetTimestamp, [267](#)
- SetTimestampVoid, [266](#)
- SetVoid, [268](#)
- ToString, [269](#)
- Value, [263](#)
- Values, [269](#)
 - Count, [271](#)
 - Iterator, [271](#)
- ValuesIterator, [271](#)
 - HasNext, [273](#)
 - Next, [273](#)
- Warning
 - gdb, [18](#)
- WeakConnectivity, [273](#)
 - AddAllEdgeTypes, [277](#), [279](#)
 - AddEdgeType, [277](#), [279](#)
 - ExcludeEdges, [278](#)
 - ExcludeNodes, [277](#)
 - GetConnectedComponents, [278](#)
 - Run, [278](#)
 - SetMaterializedAttribute, [278](#)
 - SetNodesNotVisited, [279](#)
 - WeakConnectivity, [277](#)
- WeakConnectivityDFS, [279](#)
 - AddAllEdgeTypes, [284](#)
 - AddEdgeType, [283](#)
 - ExcludeEdges, [284](#)
 - ExcludeNodes, [284](#)
 - GetConnectedComponents, [284](#)
 - SetMaterializedAttribute, [285](#)
 - SetNodesNotVisited, [285](#)
 - WeakConnectivityDFS, [283](#)
- Write
 - CSVWriter, [57](#)
 - RowWriter, [157](#)
 - TextStream, [221](#)
- WrongArgumentError, [285](#)
 - Message, [287](#)
 - NewError, [287](#)
 - SetMessage, [287](#)
 - WrongArgumentError, [287](#)
- YGraphML

gdb, [17](#)