# Sparksee

6.0.2

Generated by Doxygen 1.8.11

# Contents

**1 Module Index** **1**

1.1 Modules . . . . . 1

**2 Hierarchical Index** **1**

2.1 Class Hierarchy . . . . . 1

**3 Class Index** **5**

3.1 Class List . . . . . 5

**4 Module Documentation** **10**

4.1 Gdb . . . . . 10

4.1.1 Detailed Description . . . . . 15

4.1.2 Typedef Documentation . . . . . 15

4.1.3 Enumeration Type Documentation . . . . . 15

4.1.4 Function Documentation . . . . . 19

4.2 Io . . . . . 21

4.2.1 Detailed Description . . . . . 23

4.2.2 Function Documentation . . . . . 23

4.3 Script . . . . . 25

4.3.1 Detailed Description . . . . . 25

4.3.2 Function Documentation . . . . . 25

4.4 Algorithms . . . . . 27

4.4.1 Detailed Description . . . . . 29

4.4.2 Function Documentation . . . . . 29

**5 Class Documentation** **32**

5.1 AppError Class Reference . . . . . 32

5.1.1 Detailed Description . . . . . 33

5.1.2 Constructor & Destructor Documentation . . . . . 33

5.1.3 Member Function Documentation . . . . . 33

5.2 Attribute Class Reference . . . . . 34

5.2.1 Detailed Description . . . . . 35

5.2.2 Member Function Documentation . . . . . 35

5.3 AttributeList Class Reference . . . . . 37

5.3.1 Detailed Description . . . . . 37

5.3.2 Constructor & Destructor Documentation . . . . . 37

5.3.3 Member Function Documentation . . . . . 38

5.4 AttributeListIterator Class Reference . . . . . 38

5.4.1 Detailed Description . . . . . 39

5.4.2 Member Function Documentation . . . . . 39

5.5 AttributeStatistics Class Reference . . . . . 39

5.5.1 Detailed Description . . . . . 40

5.5.2 Member Function Documentation . . . . . 41

5.6 BooleanList Class Reference . . . . . 43

5.6.1 Detailed Description . . . . . 44

5.6.2 Constructor & Destructor Documentation . . . . . 44

5.6.3 Member Function Documentation . . . . . 44

5.7 BooleanListIterator Class Reference . . . . . 45

5.7.1 Detailed Description . . . . . 45

5.7.2 Member Function Documentation . . . . . 45

5.8 BulkLoader Class Reference . . . . . 46

5.8.1 Detailed Description . . . . . 47

5.8.2 Member Function Documentation . . . . . 47

5.9 CommunitiesSCD Class Reference . . . . . 49

5.9.1 Detailed Description . . . . . 52

5.9.2 Constructor & Destructor Documentation . . . 53
5.9.3 Member Function Documentation . . . 53
5.10 CommunityDetection Class Reference . . . 56
5.10.1 Detailed Description . . . 58
5.10.2 Constructor & Destructor Documentation . . . 58
5.10.3 Member Function Documentation . . . 58
5.11 ConnectedComponents Class Reference . . . 60
5.11.1 Detailed Description . . . 61
5.11.2 Constructor & Destructor Documentation . . . 61
5.11.3 Member Function Documentation . . . 61
5.12 Connectivity Class Reference . . . 62
5.12.1 Detailed Description . . . 65
5.12.2 Constructor & Destructor Documentation . . . 65
5.12.3 Member Function Documentation . . . 66
5.13 Context Class Reference . . . 67
5.13.1 Detailed Description . . . 69
5.13.2 Constructor & Destructor Documentation . . . 69
5.13.3 Member Function Documentation . . . 69
5.14 CSVLoader Class Reference . . . 71
5.14.1 Member Function Documentation . . . 71
5.15 CSVReader Class Reference . . . 73
5.15.1 Detailed Description . . . 74
5.15.2 Member Function Documentation . . . 74
5.16 CSVWriter Class Reference . . . 77
5.16.1 Detailed Description . . . 78
5.16.2 Member Function Documentation . . . 79
5.17 Database Class Reference . . . 80
5.17.1 Detailed Description . . . 82
5.17.2 Member Function Documentation . . . 83
5.18 DatabaseStatistics Class Reference . . . 84

5.18.1 Detailed Description . . . 84
5.18.2 Member Function Documentation . . . 84
5.19 DefaultExport Class Reference . . . 86
5.19.1 Detailed Description . . . 87
5.19.2 Member Function Documentation . . . 87
5.20 DisjointCommunities Class Reference . . . 89
5.20.1 Detailed Description . . . 90
5.20.2 Constructor & Destructor Documentation . . . 90
5.20.3 Member Function Documentation . . . 90
5.21 DisjointCommunityDetection Class Reference . . . 91
5.21.1 Detailed Description . . . 95
5.21.2 Constructor & Destructor Documentation . . . 95
5.21.3 Member Function Documentation . . . 95
5.22 EdgeData Class Reference . . . 98
5.22.1 Detailed Description . . . 98
5.22.2 Member Function Documentation . . . 98
5.23 EdgeExport Class Reference . . . 99
5.23.1 Detailed Description . . . 100
5.23.2 Member Function Documentation . . . 100
5.24 EdgeTypeExporter Class Reference . . . 102
5.24.1 Detailed Description . . . 104
5.24.2 Constructor & Destructor Documentation . . . 104
5.24.3 Member Function Documentation . . . 104
5.25 EdgeTypeLoader Class Reference . . . 107
5.25.1 Detailed Description . . . 109
5.25.2 Member Enumeration Documentation . . . 109
5.25.3 Constructor & Destructor Documentation . . . 110
5.25.4 Member Function Documentation . . . 110
5.26 EncryptionError Class Reference . . . 114
5.26.1 Detailed Description . . . 116

5.26.2 Constructor & Destructor Documentation . . . 116
5.26.3 Member Function Documentation . . . 116
5.27 Error Class Reference . . . 117
5.27.1 Detailed Description . . . 118
5.27.2 Constructor & Destructor Documentation . . . 118
5.27.3 Member Function Documentation . . . 119
5.28 Exception Class Reference . . . 119
5.28.1 Detailed Description . . . 120
5.28.2 Constructor & Destructor Documentation . . . 120
5.28.3 Member Function Documentation . . . 121
5.29 ExportManager Class Reference . . . 121
5.29.1 Detailed Description . . . 122
5.29.2 Member Function Documentation . . . 122
5.30 FileNotFoundException Class Reference . . . 125
5.30.1 Detailed Description . . . 126
5.30.2 Constructor & Destructor Documentation . . . 126
5.30.3 Member Function Documentation . . . 126
5.31 Graph Class Reference . . . 127
5.31.1 Detailed Description . . . 131
5.31.2 Member Function Documentation . . . 131
5.32 GraphExport Class Reference . . . 153
5.32.1 Detailed Description . . . 154
5.32.2 Member Function Documentation . . . 154
5.33 Handler< T > Class Template Reference . . . 154
5.33.1 Detailed Description . . . 156
5.33.2 Constructor & Destructor Documentation . . . 156
5.33.3 Member Function Documentation . . . 156
5.34 Int32List Class Reference . . . 157
5.34.1 Detailed Description . . . 157
5.34.2 Constructor & Destructor Documentation . . . 158

5.34.3 Member Function Documentation . . . 158
5.35 Int32ListIterator Class Reference . . . 159
5.35.1 Detailed Description . . . 159
5.35.2 Member Function Documentation . . . 159
5.36 IOError Class Reference . . . 160
5.36.1 Detailed Description . . . 161
5.36.2 Constructor & Destructor Documentation . . . 161
5.36.3 Member Function Documentation . . . 161
5.37 IOException Class Reference . . . 162
5.37.1 Detailed Description . . . 163
5.37.2 Constructor & Destructor Documentation . . . 163
5.37.3 Member Function Documentation . . . 164
5.38 KeyValue Class Reference . . . 164
5.39 KeyValues Class Reference . . . 164
5.39.1 Detailed Description . . . 166
5.39.2 Member Function Documentation . . . 166
5.40 KOpt Class Reference . . . 167
5.40.1 Detailed Description . . . 168
5.41 LicenseError Class Reference . . . 168
5.41.1 Detailed Description . . . 170
5.41.2 Constructor & Destructor Documentation . . . 170
5.41.3 Member Function Documentation . . . 170
5.42 SinglePairShortestPathDijkstra::FibonacciHeap::Node Struct Reference . . . 171
5.42.1 Detailed Description . . . 171
5.43 NodeExport Class Reference . . . 171
5.43.1 Detailed Description . . . 172
5.43.2 Member Function Documentation . . . 173
5.44 NodeTypeExporter Class Reference . . . 176
5.44.1 Detailed Description . . . 177
5.44.2 Constructor & Destructor Documentation . . . 177

5.44.3 Member Function Documentation . . . 178
5.45 NodeTypeLoader Class Reference . . . 181
5.45.1 Detailed Description . . . 183
5.45.2 Member Enumeration Documentation . . . 183
5.45.3 Constructor & Destructor Documentation . . . 183
5.45.4 Member Function Documentation . . . 183
5.46 NoSuchElementException Class Reference . . . 188
5.46.1 Detailed Description . . . 189
5.46.2 Constructor & Destructor Documentation . . . 189
5.46.3 Member Function Documentation . . . 189
5.47 Objects Class Reference . . . 191
5.47.1 Detailed Description . . . 193
5.47.2 Member Function Documentation . . . 193
5.48 ObjectsIterator Class Reference . . . 199
5.48.1 Detailed Description . . . 199
5.48.2 Member Function Documentation . . . 199
5.49 OIDList Class Reference . . . 200
5.49.1 Detailed Description . . . 200
5.49.2 Constructor & Destructor Documentation . . . 200
5.49.3 Member Function Documentation . . . 201
5.50 OIDListIterator Class Reference . . . 202
5.50.1 Detailed Description . . . 202
5.50.2 Member Function Documentation . . . 202
5.51 PageRank Class Reference . . . 203
5.51.1 Detailed Description . . . 204
5.51.2 Constructor & Destructor Documentation . . . 204
5.51.3 Member Function Documentation . . . 204
5.52 Platform Class Reference . . . 206
5.52.1 Detailed Description . . . 207
5.52.2 Member Function Documentation . . . 207

5.53 PlatformStatistics Class Reference . . . 207
5.53.1 Detailed Description . . . 208
5.53.2 Member Function Documentation . . . 208
5.54 Query Class Reference . . . 209
5.54.1 Detailed Description . . . 211
5.54.2 Member Function Documentation . . . 211
5.55 QueryContext Class Reference . . . 212
5.55.1 Detailed Description . . . 212
5.56 QueryException Class Reference . . . 212
5.56.1 Detailed Description . . . 214
5.56.2 Constructor & Destructor Documentation . . . 214
5.56.3 Member Function Documentation . . . 215
5.57 QueryStream Class Reference . . . 216
5.57.1 Detailed Description . . . 216
5.57.2 Member Function Documentation . . . 216
5.58 RandomWalk Class Reference . . . 217
5.58.1 Detailed Description . . . 220
5.58.2 Constructor & Destructor Documentation . . . 220
5.58.3 Member Function Documentation . . . 220
5.59 ResultSet Class Reference . . . 224
5.59.1 Detailed Description . . . 225
5.59.2 Member Function Documentation . . . 225
5.60 ResultSetList Class Reference . . . 228
5.60.1 Detailed Description . . . 228
5.60.2 Constructor & Destructor Documentation . . . 228
5.60.3 Member Function Documentation . . . 228
5.61 ResultSetListIterator Class Reference . . . 229
5.61.1 Detailed Description . . . 229
5.61.2 Member Function Documentation . . . 230
5.62 RowReader Class Reference . . . 230

5.62.1 Detailed Description . . . 231
5.62.2 Constructor & Destructor Documentation . . . 231
5.62.3 Member Function Documentation . . . 231
5.63 RowWriter Class Reference . . . 232
5.63.1 Detailed Description . . . 233
5.63.2 Constructor & Destructor Documentation . . . 233
5.63.3 Member Function Documentation . . . 234
5.64 ScriptParser Class Reference . . . 234
5.64.1 Detailed Description . . . 235
5.64.2 Member Function Documentation . . . 235
5.65 Session Class Reference . . . 237
5.65.1 Detailed Description . . . 239
5.65.2 Member Function Documentation . . . 239
5.66 ShortestPath Class Reference . . . 240
5.66.1 Detailed Description . . . 242
5.66.2 Constructor & Destructor Documentation . . . 242
5.66.3 Member Function Documentation . . . 242
5.66.4 Member Data Documentation . . . 244
5.67 SinglePairShortestPath Class Reference . . . 244
5.67.1 Detailed Description . . . 247
5.67.2 Constructor & Destructor Documentation . . . 247
5.67.3 Member Function Documentation . . . 247
5.67.4 Member Data Documentation . . . 249
5.68 SinglePairShortestPathBFS Class Reference . . . 249
5.68.1 Detailed Description . . . 252
5.68.2 Constructor & Destructor Documentation . . . 252
5.68.3 Member Function Documentation . . . 253
5.68.4 Member Data Documentation . . . 255
5.69 SinglePairShortestPathDijkstra Class Reference . . . 255
5.69.1 Detailed Description . . . 258

5.69.2 Constructor & Destructor Documentation . . . 258
5.69.3 Member Function Documentation . . . 258
5.69.4 Member Data Documentation . . . 261
5.70 SinglePairShortestPathDijkstraDynamicCost Class Reference . . . 261
5.70.1 Detailed Description . . . 262
5.70.2 Member Function Documentation . . . 262
5.71 Sparksee Class Reference . . . 262
5.71.1 Detailed Description . . . 265
5.71.2 Constructor & Destructor Documentation . . . 265
5.71.3 Member Function Documentation . . . 265
5.72 SparkseeConfig Class Reference . . . 272
5.72.1 Detailed Description . . . 276
5.72.2 Constructor & Destructor Documentation . . . 277
5.72.3 Member Function Documentation . . . 277
5.73 SparkseeProperties Class Reference . . . 292
5.73.1 Detailed Description . . . 293
5.73.2 Member Function Documentation . . . 293
5.74 StringList Class Reference . . . 295
5.74.1 Detailed Description . . . 295
5.74.2 Constructor & Destructor Documentation . . . 295
5.74.3 Member Function Documentation . . . 296
5.75 StringListIterator Class Reference . . . 296
5.75.1 Detailed Description . . . 297
5.75.2 Member Function Documentation . . . 297
5.76 StrongConnectivity Class Reference . . . 298
5.76.1 Detailed Description . . . 300
5.76.2 Constructor & Destructor Documentation . . . 301
5.76.3 Member Function Documentation . . . 301
5.77 StrongConnectivityGabow Class Reference . . . 303
5.77.1 Detailed Description . . . 306

5.77.2 Constructor & Destructor Documentation . . . 306
5.77.3 Member Function Documentation . . . 306
5.78 SystemError Class Reference . . . 308
5.78.1 Detailed Description . . . 309
5.78.2 Constructor & Destructor Documentation . . . 310
5.78.3 Member Function Documentation . . . 311
5.79 TextStream Class Reference . . . 311
5.79.1 Detailed Description . . . 313
5.79.2 Constructor & Destructor Documentation . . . 313
5.79.3 Member Function Documentation . . . 314
5.80 Traversal Class Reference . . . 315
5.80.1 Detailed Description . . . 316
5.80.2 Constructor & Destructor Documentation . . . 316
5.80.3 Member Function Documentation . . . 317
5.81 TraversalBFS Class Reference . . . 319
5.81.1 Detailed Description . . . 321
5.81.2 Constructor & Destructor Documentation . . . 321
5.81.3 Member Function Documentation . . . 321
5.82 TraversalDFS Class Reference . . . 324
5.82.1 Detailed Description . . . 326
5.82.2 Constructor & Destructor Documentation . . . 326
5.82.3 Member Function Documentation . . . 326
5.83 Type Class Reference . . . 328
5.83.1 Detailed Description . . . 329
5.83.2 Member Function Documentation . . . 329
5.84 TypeExporter Class Reference . . . 331
5.84.1 Detailed Description . . . 332
5.84.2 Constructor & Destructor Documentation . . . 332
5.84.3 Member Function Documentation . . . 333
5.85 TypeExporterEvent Class Reference . . . 336

5.85.1 Detailed Description . . . 336
5.85.2 Member Function Documentation . . . 336
5.86 TypeExporterListener Class Reference . . . 337
5.86.1 Detailed Description . . . 338
5.86.2 Constructor & Destructor Documentation . . . 338
5.86.3 Member Function Documentation . . . 338
5.87 TypeList Class Reference . . . 338
5.87.1 Detailed Description . . . 339
5.87.2 Constructor & Destructor Documentation . . . 339
5.87.3 Member Function Documentation . . . 339
5.88 TypeListIterator Class Reference . . . 340
5.88.1 Detailed Description . . . 340
5.88.2 Member Function Documentation . . . 341
5.89 TypeLoader Class Reference . . . 341
5.89.1 Detailed Description . . . 343
5.89.2 Member Enumeration Documentation . . . 343
5.89.3 Constructor & Destructor Documentation . . . 343
5.89.4 Member Function Documentation . . . 344
5.90 TypeLoaderEvent Class Reference . . . 348
5.90.1 Detailed Description . . . 349
5.91 TypeLoaderListener Class Reference . . . 349
5.91.1 Detailed Description . . . 350
5.91.2 Constructor & Destructor Documentation . . . 350
5.91.3 Member Function Documentation . . . 350
5.92 UnsupportedOperationError Class Reference . . . 350
5.92.1 Detailed Description . . . 352
5.92.2 Constructor & Destructor Documentation . . . 352
5.92.3 Member Function Documentation . . . 352
5.93 Value Class Reference . . . 353
5.93.1 Detailed Description . . . 356

5.93.2 Constructor & Destructor Documentation . . . 356
5.93.3 Member Function Documentation . . . 357
5.94 ValueArray Class Reference . . . 364
5.94.1 Detailed Description . . . 367
5.94.2 Member Function Documentation . . . 367
5.95 ValueList Class Reference . . . 377
5.95.1 Detailed Description . . . 377
5.95.2 Constructor & Destructor Documentation . . . 377
5.95.3 Member Function Documentation . . . 377
5.96 ValueListIterator Class Reference . . . 378
5.96.1 Detailed Description . . . 379
5.96.2 Member Function Documentation . . . 379
5.97 Values Class Reference . . . 380
5.97.1 Detailed Description . . . 381
5.97.2 Member Function Documentation . . . 381
5.98 ValuesIterator Class Reference . . . 382
5.98.1 Detailed Description . . . 383
5.98.2 Member Function Documentation . . . 383
5.99 WeakConnectivity Class Reference . . . 384
5.99.1 Detailed Description . . . 387
5.99.2 Constructor & Destructor Documentation . . . 387
5.99.3 Member Function Documentation . . . 387
5.100 WeakConnectivityDFS Class Reference . . . 389
5.100.1 Detailed Description . . . 393
5.100.2 Constructor & Destructor Documentation . . . 393
5.100.3 Member Function Documentation . . . 393
5.101 WrongArgumentError Class Reference . . . 395
5.101.1 Detailed Description . . . 397
5.101.2 Constructor & Destructor Documentation . . . 397
5.101.3 Member Function Documentation . . . 397

# 1 Module Index

## 1.1 Modules

Here is a list of all modules:

**Gdb** 10

**Io** 21

**Script** 25

**Algorithms** 27

# 2 Hierarchical Index

## 2.1 Class Hierarchy

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

**Attribute** 34

**AttributeList** 37

**AttributeListIterator** 38

**AttributeStatistics** 39

**BooleanList** 43

**BooleanListIterator** 45

**CommunityDetection** 56

- **DisjointCommunityDetection** 91
  - **CommunitiesSCD** 49

**ConnectedComponents** 60

**Connectivity** 62

- **StrongConnectivity** 298
  - **StrongConnectivityGabow** 303
- **WeakConnectivity** 384
  - **WeakConnectivityDFS** 389

**Context** 67

**CSVLoader** 71

**DatabaseStatistics** 84

**DisjointCommunities** 89

**EdgeData** 98

**EdgeExport** 99

**Exception** 119

**Error** 117

**AppError** 32

**EncryptionError** 114

**LicenseError** 168

**QueryException** 212

**UnsupportedOperationError** 350

**WrongArgumentError** 395

**SystemError** 308

**IOError** 160

**IOException** 162

**FileNotFoundException** 125

**NoSuchElementException** 188

**ExportManager** 121

**DefaultExport** 86

**GraphExport** 153

**Handler< T >** 154
Handler

**BulkLoader** 46

**Handler< sparksee_core::DbGraph >** 154

**Graph** 127

**Handler< sparksee_core::GraphPool >** 154

**Database** 80

**Handler< sparksee_core::GraphTextStream >** 154

**TextStream** 311

**Handler< sparksee_core::GraphValueArray >** 154

**ValueArray** 364

**Handler< sparksee_core::KeyValues >** 154

**KeyValues** 164

**Handler< sparksee_core::Objects >** 154

Objects 191

Handler< sparksee_core::Query > 154

Query 209

Handler< sparksee_core::ResultSet > 154

ResultSet 224

Handler< sparksee_core::Session > 154

Session 237

Handler< sparksee_core::SPARKSEE > 154

Sparksee 262

Handler< sparksee_core::Value > 154

Value 353

Handler< sparksee_core::Values > 154

Values 380

ValuesIterator 382

Int32List 157

Int32ListIterator 159

KeyValue 164

KOpt 167

SinglePairShortestPathDijkstra::FibonacciHeap::Node 171

NodeExport 171

ObjectsIterator 199

OIDList 200

OIDListIterator 202

PageRank 203

Platform 206

PlatformStatistics 207

QueryContext 212

QueryStream 216

ResultSetList 228

ResultSetListIterator 229

RowReader 230

CSVReader 73

RowWriter 232
CSVWriter 77
ScriptParser 234
ShortestPath 240
SinglePairShortestPath 244
SinglePairShortestPathBFS 249
SinglePairShortestPathDijkstra 255
SinglePairShortestPathDijkstraDynamicCost 261
SparkseeConfig 272
SparkseeProperties 292
StringList 295
StringListIterator 296
Traversal 315
RandomWalk 217
TraversalBFS 319
TraversalDFS 324
Type 328
TypeExporter 331
EdgeTypeExporter 102
NodeTypeExporter 176
TypeExporterEvent 336
TypeExporterListener 337
TypeList 338
TypeListIterator 340
TypeLoader 341
EdgeTypeLoader 107
NodeTypeLoader 181
TypeLoaderEvent 348
TypeLoaderListener 349
ValueList 377
ValueListIterator 378

# 3 Class Index

## 3.1 Class List

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

**AppError**
Application error class 32

**Attribute**
Attribute data class 34

**AttributeList**
Sparksee attribute identifier list 37

**AttributeListIterator**
AttributeList iterator class 38

**AttributeStatistics**
Attribute statistics class 39

**BooleanList**
Boolean list 43

**BooleanListIterator**
BooleanList iterator class 45

**BulkLoader**
Base TypeLoader class 46

**CommunitiesSCD**
CommunitiesSCD class 49

**CommunityDetection**
CommunityDetection class 56

**ConnectedComponents**
ConnectedComponents class 60

**Connectivity**
Connectivity class 62

**Context**
Context class 67

**CSVLoader** 71

**CSVReader**
CSVReader interface 73

**CSVWriter**
CSVWriter interface 77

**Database**
Database class 80

**DatabaseStatistics**
Database statistics 84

DefaultExport
Default implementation for ExportManager class 86

DisjointCommunities
DisjointCommunities class 89

DisjointCommunityDetection
DisjointCommunityDetection class 91

EdgeData
Edge data class 98

EdgeExport
Stores edge exporting values 99

EdgeTypeExporter
EdgeTypeExporter class 102

EdgeTypeLoader
EdgeTypeLoader class 107

EncryptionError
Wrong encryption arguments error class 114

Error
Error class 117

Exception
Exception class 119

ExportManager
Defines how to export a graph to an external format 121

FileNotFoundException
File not found exception class 125

Graph
Graph class 127

GraphExport
Stores the graph exporting values 153

Handler< T >
Handles a reference 154

Int32List
Sparksee 32-bit signed integer list 157

Int32ListIterator
Int32List iterator class 159

IOError
IO error class 160

IOException
IO exception class 162

KeyValue 164

KeyValues
Value set class 164

KOpt
KOpt class 167

LicenseError
License error class 168

SinglePairShortestPathDijkstra::FibonacciHeap::Node
A FibonacciHeap node structure 171

NodeExport
Stores the node exporting values 171

NodeTypeExporter
NodeTypeExporter class 176

NodeTypeLoader
NodeTypeLoader class 181

NoSuchElementException
No such element exception class 188

Objects
Object identifier set class 191

ObjectsIterator
ObjectsIterator class 199

OIDList
Sparksee object identifier list 200

OIDListIterator
OIDList iterator class 202

PageRank
PageRank class 203

Platform
Platform class 206

PlatformStatistics
Platform data and statistics 207

Query
Query class 209

QueryContext
Query context interface 212

QueryException
Query exception class 212

QueryStream
Query stream interface 216

RandomWalk
RandomWalk class 217

ResultSet
ResultSet class 224

ResultSetList
ResultSet list 228

**ResultSetListIterator**
ResultSetList iterator class 229

**RowReader**
RowReader interface 230

**RowWriter**
RowWriter interface 232

**ScriptParser**
ScriptParser 234

**Session**
Session class 237

**ShortestPath**
ShortestPath class 240

**SinglePairShortestPath**
SinglePairShortestPath class 244

**SinglePairShortestPathBFS**
SinglePairShortestPathBFS class 249

**SinglePairShortestPathDijkstra**
SinglePairShortestPathDijkstra class 255

**SinglePairShortestPathDijkstraDynamicCost**
Defines how to calculate an edge weight 261

**Sparksee**
Sparksee class 262

**SparkseeConfig**
Sparksee configuration class 272

**SparkseeProperties**
Sparksee properties file 292

**StringList**
String list 295

**StringListIterator**
StringList iterator class 296

**StrongConnectivity**
StrongConnectivity class 298

**StrongConnectivityGabow**
This class can be used to solve the problem of finding strongly connected components in a directed graph 303

**SystemError**
System error class 308

**TextStream**
TextStream class 311

**Traversal**
Traversal class 315

TraversalBFS
Breadth-First Search implementation of Traversal 319

TraversalDFS
Depth-First Search (DFS) implementation of Traversal 324

Type
Type data class 328

TypeExporter
Base TypeExporter class 331

TypeExporterEvent
Provides information about the progress of an TypeExproter instance 336

TypeExporterListener
Interface to be implemented to receive TypeExporterEvent events from a TypeExporter 337

TypeList
Sparksee type identifier list 338

TypeListIterator
TypeList iterator class 340

TypeLoader
Base TypeLoader class 341

TypeLoaderEvent
Provides information about the progress of a TypeLoader instance 348

TypeLoaderListener
Interface to be implemented to receive TypeLoaderEvent events from a TypeLoader 349

UnsupportedOperationError
Unsupported operation error class 350

Value
Value class 353

ValueArray
ValueArray class 364

ValueList
Value list 377

ValueListIterator
ValueList iterator class 378

Values
Value set class 380

ValuesIterator
Values iterator class 382

WeakConnectivity
WeakConnectivity class 384

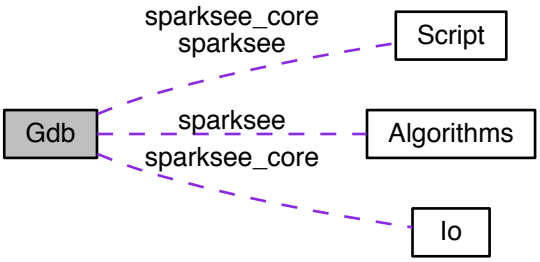
WeakConnectivityDFS
WeakConnectivityDFS class 389

WrongArgumentError
Wrong argument error class 395

# 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 Values.h

  *It contains the definition of Values class.*
- file Objects.h

  *It contains the definition of Object class.*
- file ObjectsIterator.h

  *It contains the definition of ObjectsIterator class.*
- file Query.h

  *It contains the definition of Query class.*
- file QueryContext.h

  *It contains the definition of QueryContext class.*
- file ResultSet.h

  *It contains the definition of ResultSet class.*
- file Session.h

*It contains the definition of Session class.*

- file Sparksee.h

  *It contains the declaration of Sparksee class.*
- file SparkseeConfig.h

  *It contains the declaration of SparkseeProperties and SparkseeConfig classes.*
- file Stream.h

  *It contains the definition of stream classes.*
- file Value.h

  *It contains the definitio of Value class.*
- file ValueArray.h

  *It contains the definition of ValueArray classes.*
- 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 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 QueryException

  *Query exception class.*

- class EncryptionError

  *Wrong encryption arguments 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 KeyValue
- class KeyValues

  *Value set class.*

- class Objects
  *Object identifier set class.*
- class ObjectsIterator
  *ObjectsIterator class.*
- class QueryStream
  *Query stream interface.*
- class Query
  *Query class.*
- class QueryContext
  *Query context interface.*
- class ResultSet
  *ResultSet class.*
- class ResultSetList
  *ResultSet list.*
- class ResultSetListIterator
  *ResultSetList iterator class.*
- class Session
  *Session class.*
- class Sparksee
  *Sparksee class.*
- class SparkseeProperties
  *Sparksee properties file.*
- class SparkseeConfig
  *Sparksee configuration class.*
- class TextStream
  *TextStream class.*
- class Value
  *Value class.*
- class ValueList
  *Value list.*
- class ValueListIterator
  *ValueList iterator class.*
- class ValueArray
  *ValueArray class.*
- class Values
  *Value set class.*
- class ValuesIterator
  *Values iterator class.*


### Macros


- #define BEGIN_SPARKSEE_NAMESPACE namespace sparksee {
  *Begining 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 {
  *Begining 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 {

*Begining 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 {

  *Begining 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 {

  *Begining 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 {

  *Begining macro for the sparksee::test namespace.*
- #define END_SPARKSEE_TEST_NAMESPACE END_SPARKSEE_NAMESPACE }

  *Ending macro for the sparksee::test namespace.*
- #define BEGIN_SPARKSEE_QUERY_STREAMS_NAMESPACE BEGIN_SPARKSEE_NAMESPACE namespace query_streams {

  *Begining macro for the sparksee::algorithms namespace.*
- #define END_SPARKSEE_QUERY_STREAMS_NAMESPACE END_SPARKSEE_NAMESPACE }

  *Ending macro for the sparksee::query_streams 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.*
- typedef void(∗ SPARKSEE_UNRECOVERABLE_ERR_CALLBACK) (UnrecoverableError error)

  *Signature of callback functions to call after UnrecoverableErrors are triggered.*

#### Enumerations

#### 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.*

#### Variables

- BEGIN_SPARKSEE_GDB_NAMESPACE typedef unsigned char byte_t

  *Byte type.*

### 4.1.1 Detailed Description

### 4.1.2 Typedef Documentation

#### 4.1.2.1 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.

#### 4.1.2.2 typedef void(* SPARKSEE_UNRECOVERABLE_ERR_CALLBACK) (UnrecoverableError error)

Signature of callback functions to call after UnrecoverableErrors are triggered.

**Parameters**

| | |
|---|---|
| *The* | type of UnrecoverableError that triggered the callback |

### 4.1.3 Enumeration Type Documentation

#### 4.1.3.1 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.

#### 4.1.3.2 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.

#### 4.1.3.3 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.

#### 4.1.3.4 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.

#### 4.1.3.5 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").

#### 4.1.3.6 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.

#### 4.1.3.7 enum MissingEndpoint

The policy to follow whenever and edge endpoint is missing during a loading.

**Enumerator**

***IsError*** Throw an error.

***Create*** Create the endpoint.

***Ignore*** Ignore the edge.

#### 4.1.3.8 enum NodeShape

Node shape.

**Author**

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

**Enumerator**

***Box*** Box shape.

***Round*** Round shape.

#### 4.1.3.9 enum ObjectType

Object type enumeration.

**Author**

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

**Enumerator**

***Node*** Node object type.

***Edge*** Edge object type.

#### 4.1.3.10 enum Order

Order enumeration.

**Author**

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

**Enumerator**

***Ascendent*** From lower to higher.

***Descendent*** From higher to lower.

#### 4.1.3.11 enum QueryLanguage

The supported query languages.

**Enumerator**

***SparkseeAlgebra*** The internal Sparksee Algebra.

***SparkseeCypher*** The Sparksee Cypher Language inspired by the OpenCypher Query Language.

#### 4.1.3.12 enum UnrecoverableError

Enum representing unrecoverable errors.

**Enumerator**

***SystemCallError*** Error triggered due to an unrecoverable error detected during a system call in the Sparksee core.

***InvalidChecksum*** Error triggered after detecting an invalid checksum during reading data from an I/O device.

### 4.1.4 Function Documentation

#### 4.1.4.1 std::wostream& operator<< ( std::wostream & *wostrm*, const enum DataType & *dt* )

Easy STL printing operator redefinition.

It allows to do: ... << sparksee::gdb::String << ...

**Parameters**

| | |
|---|---|
| *wostrm* | A widechar oputput stream |
| *dt* | [in] An DataType. |

**Returns**

Returns the widechar output stream with the DataType written.

#### 4.1.4.2 std::wostream& operator<< ( std::wostream & *wostrm*, const enum AttributeKind & *ak* )

Easy STL printing operator redefinition.

It allows to do: ... << sparksee::gdb::Basic << ...

**Parameters**

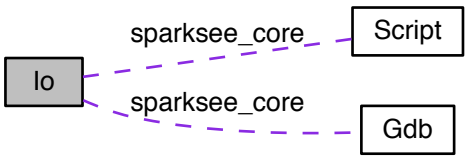
| | |
|---|---|
| *wostrm* | A widechar oputput stream |
| *ak* | [in] An attribute kind. |

**Returns**

Returns the widechar output stream with the AttributeKind written.

## 4.2 Io

Collaboration diagram for Io:




### Files

- file TypeLoader.h

  *It contains the definition of TypeLoader classes.*
- 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 BulkLoader

  *Base TypeLoader class.*
- 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)

  *Output Stream TypeLoaderEvent operator definition.*
- sparksee::gdb::type_t TypeLoaderEvent::GetTypeId () const

  *Gets the type identifier.*
- sparksee::gdb::int64_t TypeLoaderEvent::GetCount () const

  *Gets the current number of objects created.*
- sparksee::gdb::int32_t TypeLoaderEvent::GetPhase () const

  *Gets the current phase.*
- sparksee::gdb::int32_t TypeLoaderEvent::GetTotalPhases () const

  *Gets the total number of phases.*
- sparksee::gdb::int32_t TypeLoaderEvent::GetPartition () const

  *Gets the current partition.*
- sparksee::gdb::int32_t TypeLoaderEvent::GetTotalPartitions () const

  *Gets the total number of partitions.*
- sparksee::gdb::int32_t TypeLoaderEvent::GetTotalPartitionSteps () const

  *Gets the total number of steps in the current partition.*
- sparksee::gdb::bool_t TypeLoaderEvent::IsLast () const

  *Gets if this is the last event or not.*

#### 4.2.1 Detailed Description

#### 4.2.2 Function Documentation

##### 4.2.2.1 sparksee::gdb::int64_t TypeLoaderEvent::GetCount ( ) const [inline]

Gets the current number of objects created.

**Returns**

The current number of objects created.

##### 4.2.2.2 sparksee::gdb::int32_t TypeLoaderEvent::GetPartition ( ) const [inline]

Gets the current partition.

**Returns**

The current partition.

##### 4.2.2.3 sparksee::gdb::int32_t TypeLoaderEvent::GetPhase ( ) const [inline]

Gets the current phase.

**Returns**

The current phase.

##### 4.2.2.4 sparksee::gdb::int32_t TypeLoaderEvent::GetTotalPartitions ( ) const [inline]

Gets the total number of partitions.

**Returns**

The total number of partitions.

##### 4.2.2.5 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.

##### 4.2.2.6 sparksee::gdb::int32_t TypeLoaderEvent::GetTotalPhases ( ) const [inline]

Gets the total number of phases.

**Returns**

The total number of phases.

#### 4.2.2.7 sparksee::gdb::type_t TypeLoaderEvent::GetTypeId ( ) const [inline]

Gets the type identifier.

**Returns**

The type identifier.

#### 4.2.2.8 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.

References END_SPARKSEE_IO_NAMESPACE.

#### 4.2.2.9 std::wostream& operator<< ( std::wostream & *wostrm,* const TypeLoaderEvent & *ev* )

Output Stream TypeLoaderEvent operator definition.

**Parameters**

| | |
|---|---|
| *wostrm* | The stream to print to |
| *ev* | The TypeLoaderEvent to print |

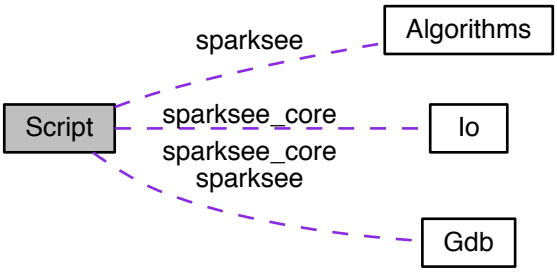
**Returns**

A reference to the stream

Referenced by TypeLoaderEvent::~TypeLoaderEvent().

## 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 Detailed Description

#### 4.3.2 Function Documentation

##### 4.3.2.1 std::wostream& operator<< ( std::wostream & *wostrm,* const enum ScriptParserState & *state* )

Easy STL printing operator redefinition.

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

**Parameters**

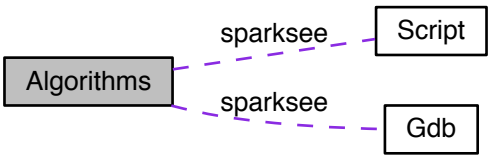
| | |
|---|---|
| *wostrm* | A widechar oputput stream |
| *state* | [in] A ScriptParserState. |

**Returns**

Returns the widechar output stream with the ScriptParserState written.

## 4.4 Algorithms

Collaboration diagram for Algorithms:




### Files

- file CommunitiesSCD.h

  *It contains the definition of CommunitiesSCD class.*
- file CommunityDetection.h

  *It contains the definition of CommunityDetection class.*
- 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 DisjointCommunities.h

  *It contains the definition of DisjointCommunities class.*
- file DisjointCommunityDetection.h

  *It contains the definition of DisjointCommunityDetection class.*
- file KOpt.h

  *It contains the definition of KOpt class.*
- file PageRank.h

  *It contains the definition of PageRank class.*
- file RandomWalk.h

  *It contains the definition of RandomWalk 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 CommunitiesSCD

  *CommunitiesSCD class.*
- class CommunityDetection

  *CommunityDetection class.*
- class ConnectedComponents

  *ConnectedComponents class.*
- class Connectivity

  *Connectivity class.*
- class Context

  *Context class.*
- class DisjointCommunities

  *DisjointCommunities class.*
- class DisjointCommunityDetection

  *DisjointCommunityDetection class.*
- class KOpt

  *KOpt class.*
- class PageRank

  *PageRank class.*
- class RandomWalk

  *RandomWalk class.*
- class ShortestPath

  *ShortestPath class.*
- class SinglePairShortestPath

  *SinglePairShortestPath class.*
- class SinglePairShortestPathBFS

  *SinglePairShortestPathBFS class.*
- class SinglePairShortestPathDijkstraDynamicCost

  *Defines how to calculate an edge weight.*
- 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.*

#### Functions

- KOpt::KOpt (sparksee::gdb::Session &session)

  *Creates a new instance.*
- KOpt::KOpt (sparksee::gdb::Session &session, sparksee::gdb::OIDList &tour)

  *Creates a new instance.*
- virtual KOpt::~KOpt ()

  *Destructor.*
- void KOpt::AddNodeType (sparksee::gdb::type_t type) throw (sparksee::gdb::Error)

  *Allows for traversing nodes of the given type.*
- void KOpt::AddAllNodeTypes ()

  *Allows for traversing all node types of the graph.*
- void KOpt::AddEdgeType (sparksee::gdb::type_t type, sparksee::gdb::EdgesDirection dir) throw (sparksee↩
  ::gdb::Error)

  *Allows for traversing edges of the given type.*
- void KOpt::AddAllEdgeTypes (sparksee::gdb::EdgesDirection dir)

  *Allows for traversing all edge types of the graph.*
- void KOpt::SetEdgeWeightAttributeType (sparksee::gdb::attr_t attr) throw (sparksee::gdb::Error)

  *Sets the attribute to use as edge weight.*
- double KOpt::GetCurrentCost () const

  *Returns tour cost.*
- sparksee::gdb::OIDList * KOpt::GetCurrentTour ()

  *Returns tour as a list of nodes.*
- void KOpt::SetCurrentTour (sparksee::gdb::OIDList &tour)

  *Sets current tour as a list of nodes.*
- void KOpt::SetMaxIterations (sparksee::gdb::int64_t maxIterations)

  *Sets maximum number of iterations.*
- void KOpt::SetTimeLimit (sparksee::gdb::int64_t maxTime)

  *Limits execution time.*
- void KOpt::RunTwoOpt ()

  *Runs 2-Opt local search.*
- void KOpt::RunThreeOpt ()

  *Runs 3-Opt local search.*

### 4.4.1 Detailed Description

### 4.4.2 Function Documentation

#### 4.4.2.1 void KOpt::AddAllEdgeTypes ( sparksee::gdb::EdgesDirection *dir* )

Allows for traversing all edge types of the graph.

**Parameters**

| | |
|---|---|
| *dir* | [in] Edge direction. |

#### 4.4.2.2 void KOpt::AddEdgeType ( sparksee::gdb::type_t *type,* sparksee::gdb::EdgesDirection *dir* ) throw sparksee::gdb::Error)

Allows for traversing edges of the given type.

If the edge type was already added, the existing direction is overwritten

**Parameters**

| | |
|---|---|
| *type* | [in] Edge type. |
| *dir* | [in] Edge direction. |

#### 4.4.2.3 void KOpt::AddNodeType ( sparksee::gdb::type_t *type* ) throw sparksee::gdb::Error)

Allows for traversing nodes of the given type.

**Parameters**

| | |
|---|---|
| *type* | [in] Node type. |

**Exceptions**

| | |
|---|---|
| *sparksee::gdb::Error* | |

#### 4.4.2.4 KOpt::KOpt ( sparksee::gdb::Session & *session* ) [explicit]

Creates a new instance.

**Parameters**

| | |
|---|---|
| *session* | [in] Session to get the graph from and perform algorithm. |

#### 4.4.2.5 KOpt::KOpt ( sparksee::gdb::Session & *session,* sparksee::gdb::OIDList & *tour* )

Creates a new instance.

**Parameters**

| | |
|---|---|
| *session* | [in] Session to get the graph from and perform algorithm. |
| *tour* | [in] Initial tour that needs to be improved. |

#### 4.4.2.6 void KOpt::SetCurrentTour ( sparksee::gdb::OIDList & *tour* )

Sets current tour as a list of nodes.

**Parameters**

| | |
|---|---|
| *tour* | [in] Initial tour that needs to be improved. |

#### 4.4.2.7 void KOpt::SetEdgeWeightAttributeType ( sparksee::gdb::attr_t *attr* ) throw sparksee::gdb::Error)

Sets the attribute to use as edge weight.

If the multiple edge are set for traversal, this attribute must be of type GLOBAL_TYPE or EDGES_TYPE. Additionally, the attribute must be of type Double.

**Parameters**

| | |
|---|---|
| *attr* | [in] The attribute type to use as a weight. Default: InvalidAttribute |

**Exceptions**

| | |
|---|---|
| *sparksee::gdb::Error* | |

#### 4.4.2.8 void KOpt::SetMaxIterations ( sparksee::gdb::int64_t *maxIterations* )

Sets maximum number of iterations.

By default the algorithm will run until no improvement can be made in the current tour.

**Parameters**

| | |
|---|---|
| *maxIterations* | [in] Maximum number of iterations |

#### 4.4.2.9 void KOpt::SetTimeLimit ( sparksee::gdb::int64_t *maxTime* )

Limits execution time.

**Parameters**

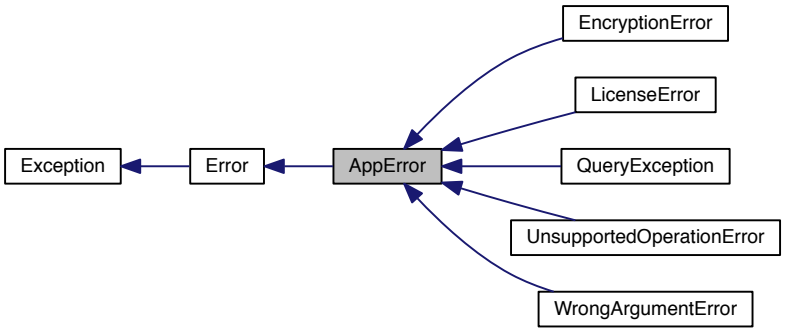
| | |
|---|---|
| *maxTime* | [in] Time limit in miliseconds |

# 5 Class Documentation

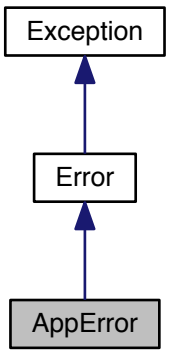
## 5.1 AppError Class Reference

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 void ThrowError (int32_t coreErrorCode)

  *Throws 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 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.

#### 5.1.3.2 void Exception::SetMessage ( const std::string & *mess* ) [inherited]

Sets the message of the exception.

**Parameters**

| | |
|---|---|
| *mess* | [in] Message. |

#### 5.1.3.3 static void Error::ThrowError ( int32_t *coreErrorCode* ) [static],[inherited]

Throws 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 will throw an Error instance of an specific Error subclass instance.

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.*
- bool_t IsArrayAttribute () const
  *Check if it's an array attribute.*
- int32_t GetArraySize () const
  *Gets the number of elements in the array.*

#### 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 int32_t Attribute::GetArraySize ( ) const [inline]

Gets the number of elements in the array.

#### Returns

The size of the array

#### 5.2.2.2 int64_t Attribute::GetCount ( ) const [inline]

Gets the number of non-NULL values.

#### Returns

The number of non-NULL values.

#### 5.2.2.3 DataType Attribute::GetDataType ( ) const [inline]

Gets the data type.

#### Returns

The DataType.

#### 5.2.2.4 attr_t Attribute::GetId ( ) const [inline]

Gets the Sparksee attribute identifier.

**Returns**

The Sparksee attribute identifier.

#### 5.2.2.5 AttributeKind Attribute::GetKind ( ) const [inline]

Gets the attribute kind.

**Returns**

The AttributeKind.

#### 5.2.2.6 const std::wstring& Attribute::GetName ( ) const [inline]

Gets the unique attribute name.

**Returns**

The unique attribute name.

#### 5.2.2.7 int64_t Attribute::GetSize ( ) const [inline]

Gets the number of different values.

**Returns**

The number of different values.

#### 5.2.2.8 type_t Attribute::GetTypeId ( ) const [inline]

Gets the Sparksee type identifier.

**Returns**

The Sparksee type identifier.

#### 5.2.2.9 bool_t Attribute::IsArrayAttribute ( ) const [inline]

Check if it's an array attribute.

**Returns**

True if it's an array attribute, or false otherwise.

#### 5.2.2.10 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 void AttributeList::Add ( attr_t *attr* ) [inline]

Adds a Sparksee attribute identifier at the end of the list.

**Parameters**

| | |
|---|---|
| *attr* | [in] Sparksee attribute identifier. |

##### 5.3.3.2 int32_t AttributeList::Count ( ) const [inline]

Number of elements in the list.

**Returns**

Number of elements in the list.

##### 5.3.3.3 AttributeListIterator* AttributeList::Iterator ( )

Gets a new AttributeListIterator.

**Returns**

AttributeListIterator instance.

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 bool_t AttributeListIterator::HasNext ( ) [inline]

Gets if there are more elements.

#### Returns

TRUE if there are more elements, FALSE otherwise.

#### 5.4.2.2 attr_t AttributeListIterator::Next ( ) [inline]

Moves to the next element.

#### Returns

The next element.

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

- Graph_data.h

## 5.5 AttributeStatistics Class Reference

Attribute statistics class.

#### 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 get↩ AttributeStatistics 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 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.

##### 5.5.2.2 int64_t AttributeStatistics::GetDistinct ( ) const [inline]

Gets the number of distinct values (BASIC statistics).

**Returns**

The number of distinct values.

##### 5.5.2.3 const Value& AttributeStatistics::GetMax ( ) const [inline]

Gets the maximum existing value (BASIC statistics).

**Returns**

The maximum existing value.

##### 5.5.2.4 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.

##### 5.5.2.5 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.

#### 5.5.2.6 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.

#### 5.5.2.7 const Value& AttributeStatistics::GetMin ( ) const [inline]

Gets the minimum existing value (BASIC statistics).

**Returns**

The minimum existing value.

#### 5.5.2.8 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.

#### 5.5.2.9 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 int64_t AttributeStatistics::GetNull ( ) const [inline]

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

**Returns**

The number of objects NULL a Value.

#### 5.5.2.12 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.

#### 5.5.2.13 double64_t AttributeStatistics::GetVariance ( ) const [inline]

Gets the variance.

It is computed just for numerical attributes.

**Returns**

The variance.

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 value)
  *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 void BooleanList::Add ( sparksee::gdb::bool_t *value* ) [inline]

Adds a Boolean at the end of the list.

**Parameters**

| | |
|---|---|
| *value* | [in] Boolean. |

#### 5.6.3.2 int32_t BooleanList::Count ( ) const [inline]

Number of elements in the list.

**Returns**

Number of elements in the list.

#### 5.6.3.3 BooleanListIterator* BooleanList::Iterator ( )

Gets a new BooleanListIterator.

**Returns**

BooleanListIterator instance.

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 bool_t BooleanListIterator::HasNext ( ) [inline]

Gets if there are more elements.

**Returns**

TRUE if there are more elements, FALSE otherwise.

#### 5.7.2.2 sparksee::gdb::bool_t BooleanListIterator::Next ( ) [inline]

Moves to the next element.

**Returns**

The next element.

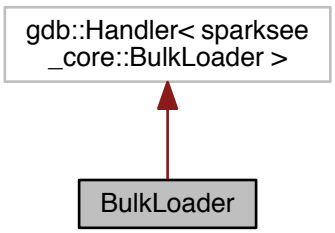
The documentation for this class was generated from the following file:

- Graph_data.h

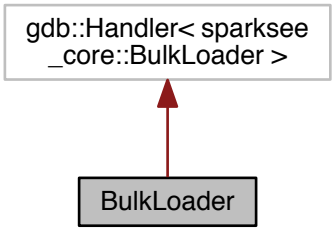
## 5.8 BulkLoader Class Reference

Base TypeLoader class.

Inheritance diagram for BulkLoader:




Collaboration diagram for BulkLoader:



**Public Member Functions**

- void SetLogError (const std::wstring &path) throw (sparksee::gdb::IOException)

  *Sets a log error file.*
- virtual ~BulkLoader ()

  *Destructor.*
- virtual void Run () throw (sparksee::gdb::IOException, sparksee::gdb::Error)

  *Run the loader.*
- void SetFilename (const std::wstring &filename)

  *Sets the input data source.*
- void SetLocale (const std::wstring &localeStr)

  *Sets the locale that will be used to read the data.*
- void SetType (sparksee::gdb::type_t type)

  *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 &timestampFormat)

  *Sets a specific timestamp format.*

### 5.8.1 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

### 5.8.2 Member Function Documentation

#### 5.8.2.1 void BulkLoader::SetAttributePositions ( sparksee::gdb::Int32List & *attrsPos* )

Sets the list of attribute positions.

**Parameters**

| | |
|---|---|
| *attrsPos* | [in] Attribute positions (column index >=0). |

#### 5.8.2.2 void BulkLoader::SetAttributes ( sparksee::gdb::AttributeList & *attrs* )

Sets the list of Attributes.

**Parameters**

| | |
|---|---|
| *attrs* | [in] Attribute identifiers to be loaded |

#### 5.8.2.3 void BulkLoader::SetFilename ( const std::wstring & *filename* )

Sets the input data source.

**Parameters**

| | |
|---|---|
| *rr* | [in] Input RowReader. |

#### 5.8.2.4 void BulkLoader::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. |

#### 5.8.2.5 void BulkLoader::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.

**Parameters**

| | |
|---|---|
| *path* | [in] The path to the error log file. |

**Exceptions**

| | |
|---|---|
| *IOException* | If bad things happen opening the file. |

#### 5.8.2.6 void BulkLoader::SetTimestampFormat ( const std::wstring & *timestampFormat* )

Sets a specific timestamp format.

**Parameters**

| | |
|---|---|
| *timestampFormat* | [in] A string with the timestamp format definition. |

#### 5.8.2.7 void BulkLoader::SetType ( sparksee::gdb::type_t *type* )

Sets the type to be loaded.

**Parameters**

| | |
|---|---|
| *type* | [in] Type identifier. |

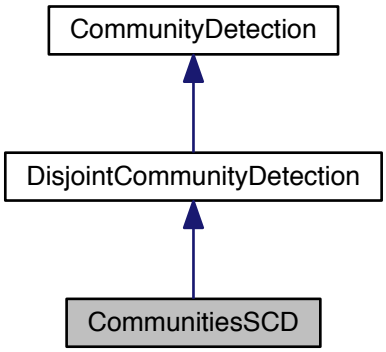
The documentation for this class was generated from the following file:

- BulkLoader.h

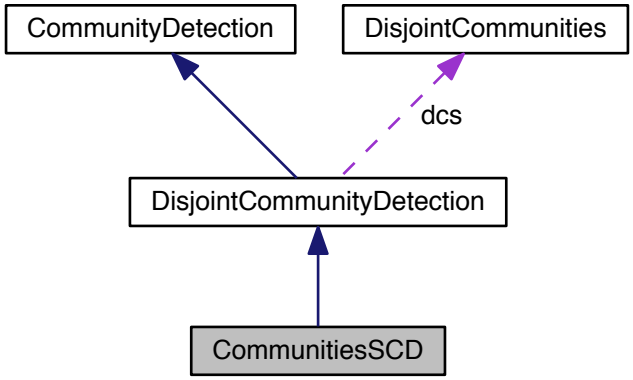
## 5.9 CommunitiesSCD Class Reference

CommunitiesSCD class.

Inheritance diagram for CommunitiesSCD:



Collaboration diagram for CommunitiesSCD:




**Public Member Functions**

- CommunitiesSCD (sparksee::gdb::Session &session)

  *Creates a new instance of CommunitiesSCD.*
- virtual ~CommunitiesSCD ()

  *Destructor.*
- void SetLookAhead (sparksee::gdb::int32_t lookahead)

  *Sets the size of the lookahead iterations to look (5 by default).*
- void Run ()

  *Executes the algorithm.*
- virtual void AddEdgeType (sparksee::gdb::type_t type)

  *Allows connectivity through edges of the given type.*
- virtual void AddAllEdgeTypes ()

  *Allows connectivity through all edge types of the graph.*
- DisjointCommunities * GetCommunities ()

  *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 disjoint communities found while executing this algorithm.*
- virtual void AddNodeType (sparksee::gdb::type_t type)

  *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.*
- virtual void IncludeNodes (sparksee::gdb::Objects &nodes)

  *Set additional nodes that can be used.*
- virtual void IncludeEdges (sparksee::gdb::Objects &edges)

  *Set additional edges that can be used.*

### 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 type, sparksee::gdb::EdgesDirection direction)

  *Allows connectivity through edges of the given type.*
- void AddAllEdgeTypes (sparksee::gdb::EdgesDirection direction)

  *Allows connectivity through all edge types of the graph.*
- void SetCommunity (sparksee::gdb::oid_t idNode)

  *Assigns the current community to the given node.*
- void AssertNotCommunityAttribute (const std::wstring &attributeName)

  *Check that the given attribute name is not already in use.*
- void CreateGlobalPersistentAttribute (const std::wstring &attributeName)

  *Set a new persistent global attribute to store the disjoint communities information.*
- void CreateGlobalTransientAttribute ()

  *Set a new temporary global attribute to store the disjoint communities information.*
- void RemoveGlobalAttribute ()

  *Removes the global attribute where the disjoint communities information is stored.*
- 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 SetNodesNotVisited ()

  *Set all the selected nodes in nodesNotVisited.*
- void AssertComputed ()

  *Check that the communities 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.*
- sparksee::gdb::bool_t IsNodeTypeAllowed (sparksee::gdb::oid_t nodeId)

  *Check if the given node has an allowed type.*
- sparksee::gdb::bool_t IsEdgeTypeAllowed (sparksee::gdb::oid_t edgeId)

  *Check if the given edge has an allowed type.*
- sparksee::gdb::bool_t IsNodeAllowed (sparksee::gdb::oid_t nodeId)

  *Check if the given node is allowed (by type or by explicit inclusion).*
- sparksee::gdb::bool_t IsEdgeAllowed (sparksee::gdb::oid_t edgeId)

  *Check if the given edge is allowed (by type or by explicit inclusion).*
- 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.*
- sparksee::gdb::bool_t IsNodeIncluded (sparksee::gdb::oid_t node)

  *Check if the given node is EXPLICITLY allowed using the include set.*
- sparksee::gdb::bool_t IsEdgeIncluded (sparksee::gdb::oid_t edge)

  *Check if the given edge is EXPLICITLY allowed using the include set.*

#### Protected Attributes

- sparksee::gdb::attr_t attrCommunity
  *common attribute where the connected component information is stored.*
- std::wstring attrCommunityName
  *name of the common attribute where the connected component information is stored.*
- sparksee::gdb::int64_t actualCommunity
  *Current community identifier.*
- sparksee::gdb::bool_t matResults
  *Materialized results.*
- DisjointCommunities * dcs
  *The calculated communities information.*
- 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::Objects * nodesNotVisited
  *Identifiers of the nodes not visited.*
- 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.*
- sparksee::gdb::Objects * includedNodes
  *The set of explicitly included nodes.*
- sparksee::gdb::Objects * includedEdges
  *The set of explicitly included edges.*

### 5.9.1 Detailed Description

CommunitiesSCD class.

Implementation of the community detection algorithm "Scalable Community Detection" based on the paper "High quality, scalable and parallel community detection for large real graphs" by Arnau Prat-Perez, David Dominguez-Sal, Josep-Lluis Larriba-Pey - WWW 2014.

The purpose of this algorithm is to find disjoint communities in an **undirected** graph or in a directed graph which will be considered as an undirected one.

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 DisjointCommunities class using the getCommunities method.

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

#### Author

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

#### 5.9.2 Constructor & Destructor Documentation

##### 5.9.2.1 CommunitiesSCD::CommunitiesSCD ( sparksee::gdb::Session & *session* )

Creates a new instance of CommunitiesSCD.

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 communities.

**Parameters**

| | |
|---|---|
| *session* | [in] Session to get the graph from and calculate the communities |

#### 5.9.3 Member Function Documentation

##### 5.9.3.1 virtual void DisjointCommunityDetection::AddAllEdgeTypes ( ) [virtual], [inherited]

Allows connectivity through all edge types of the graph.

The edges can be used in Any direction.

##### 5.9.3.2 void CommunityDetection::AddAllEdgeTypes ( **sparksee::gdb::EdgesDirection** *direction* ) [protected], [inherited]

Allows connectivity through all edge types of the graph.

**Parameters**

| | |
|---|---|
| *d* | [in] Edge direction. |

##### 5.9.3.3 virtual void DisjointCommunityDetection::AddEdgeType ( **sparksee::gdb::type_t** *type* ) [virtual], [inherited]

Allows connectivity through edges of the given type.

The edges can be used in Any direction.

**Parameters**

| | |
|---|---|
| *type* | [in] Edge type. |

##### 5.9.3.4 void CommunityDetection::AddEdgeType ( **sparksee::gdb::type_t** *type,* **sparksee::gdb::EdgesDirection** *direction* ) [protected], [inherited]

Allows connectivity through edges of the given type.

**Parameters**

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

#### 5.9.3.5 virtual void CommunityDetection::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. |

#### 5.9.3.6 virtual void CommunityDetection::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. |

#### 5.9.3.7 DisjointCommunities* DisjointCommunityDetection::GetCommunities ( ) [inherited]

Returns the results generated by the execution of the algorithm.

These results contain information related to the disjoint communities 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 DisjointCommunities which contain information related to the disjoint communities found.

#### 5.9.3.8 virtual void CommunityDetection::IncludeEdges ( sparksee::gdb::Objects & *edges* ) [virtual], [inherited]

Set additional edges that can be used.

This will replace any previously specified set of include edges. Using this optional method adds valid edges to the edges of any edge type explicitly set as a valid type. Should only be used to include specific small sets of edges because it's less efficient than just using an edge type. For any edge to be used, both nodes must be also valid.

**Parameters**

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

#### 5.9.3.9 virtual void CommunityDetection::IncludeNodes ( sparksee::gdb::Objects & *nodes* ) [virtual], [inherited]

Set additional nodes that can be used.

This will replace any previously specified set of include nodes. Using this optional method adds valid nodes to the nodes of any node type explicitly set as a valid type. Should only be used to include specific small sets of nodes because it's less efficient than just using a node type.

**Parameters**

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

#### 5.9.3.10 void CommunitiesSCD::SetLookAhead ( sparksee::gdb::int32_t *lookahead* )

Sets the size of the lookahead iterations to look (5 by default).

**Parameters**

| | |
|---|---|
| *lookahead* | [in] Number of iterations. It must be positive or zero. |

#### 5.9.3.11 void DisjointCommunityDetection::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 disjoint communities 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 DisjointCommunities 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 disjoint communities 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. |

#### 5.9.3.12 void CommunityDetection::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.

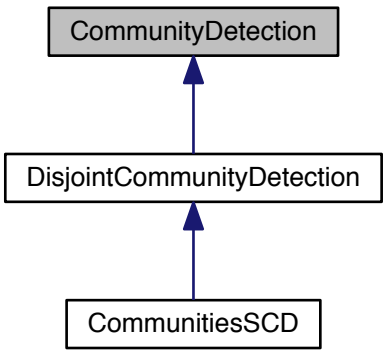
The documentation for this class was generated from the following file:

- CommunitiesSCD.h

## 5.10 CommunityDetection Class Reference

CommunityDetection class.

Inheritance diagram for CommunityDetection:




### Public Member Functions

- virtual ~CommunityDetection ()
  *Destructor.*
- virtual void AddNodeType (sparksee::gdb::type_t type)
  *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.*
- virtual void IncludeNodes (sparksee::gdb::Objects &nodes)
  *Set additional nodes that can be used.*
- virtual void IncludeEdges (sparksee::gdb::Objects &edges)
  *Set additional edges that can be used.*
- virtual void Run ()=0
  *Runs the algorithm in order to find the connected components.*

### 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

- CommunityDetection (sparksee::gdb::Session &s)

  *Creates a new instance of CommunityDetection.*
- void AddEdgeType (sparksee::gdb::type_t type, sparksee::gdb::EdgesDirection direction)

  *Allows connectivity through edges of the given type.*
- void AddAllEdgeTypes (sparksee::gdb::EdgesDirection direction)

  *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 SetNodesNotVisited ()

  *Set all the selected nodes in nodesNotVisited.*
- void AssertComputed ()

  *Check that the communities 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.*
- sparksee::gdb::bool_t IsNodeTypeAllowed (sparksee::gdb::oid_t nodeId)

  *Check if the given node has an allowed type.*
- sparksee::gdb::bool_t IsEdgeTypeAllowed (sparksee::gdb::oid_t edgeId)

  *Check if the given edge has an allowed type.*
- sparksee::gdb::bool_t IsNodeAllowed (sparksee::gdb::oid_t nodeId)

  *Check if the given node is allowed (by type or by explicit inclusion).*
- sparksee::gdb::bool_t IsEdgeAllowed (sparksee::gdb::oid_t edgeId)

  *Check if the given edge is allowed (by type or by explicit inclusion).*
- 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.*
- sparksee::gdb::bool_t IsNodeIncluded (sparksee::gdb::oid_t node)

  *Check if the given node is EXPLICITLY allowed using the include set.*
- sparksee::gdb::bool_t IsEdgeIncluded (sparksee::gdb::oid_t edge)

  *Check if the given edge is EXPLICITLY allowed using the include set.*

### 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::Objects * nodesNotVisited

*Identifiers of the nodes not visited.*

- 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.*
- sparksee::gdb::Objects * includedNodes

  *The set of explicitly included nodes.*
- sparksee::gdb::Objects * includedEdges

  *The set of explicitly included edges.*

### 5.10.1 Detailed Description

CommunityDetection class.

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

### 5.10.2 Constructor & Destructor Documentation

#### 5.10.2.1 CommunityDetection::CommunityDetection ( sparksee::gdb::Session & *s* ) [protected]

Creates a new instance of CommunityDetection.

**Parameters**

| | |
|---|---|
| *s* | [in] Session to get the graph from and calculate the communities |

### 5.10.3 Member Function Documentation

#### 5.10.3.1 void CommunityDetection::AddAllEdgeTypes ( sparksee::gdb::EdgesDirection *direction* ) [protected]

Allows connectivity through all edge types of the graph.

**Parameters**

| | |
|---|---|
| *d* | [in] Edge direction. |

#### 5.10.3.2 void CommunityDetection::AddEdgeType ( sparksee::gdb::type_t *type,* sparksee::gdb::EdgesDirection *direction* ) [protected]

Allows connectivity through edges of the given type.

**Parameters**

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

#### 5.10.3.3 virtual void CommunityDetection::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. |

#### 5.10.3.4 virtual void CommunityDetection::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. |

#### 5.10.3.5 virtual void CommunityDetection::IncludeEdges ( sparksee::gdb::Objects & *edges* ) [virtual]

Set additional edges that can be used.

This will replace any previously specified set of include edges. Using this optional method adds valid edges to the edges of any edge type explicitly set as a valid type. Should only be used to include specific small sets of edges because it's less efficient than just using an edge type. For any edge to be used, both nodes must be also valid.

**Parameters**

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

#### 5.10.3.6 virtual void CommunityDetection::IncludeNodes ( sparksee::gdb::Objects & *nodes* ) [virtual]

Set additional nodes that can be used.

This will replace any previously specified set of include nodes. Using this optional method adds valid nodes to the nodes of any node type explicitly set as a valid type. Should only be used to include specific small sets of nodes because it's less efficient than just using a node type.

**Parameters**

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

#### 5.10.3.7 virtual void CommunityDetection::Run ( ) [pure virtual]

Runs the algorithm in order to find the connected components.

This method can be called only once.

Implemented in DisjointCommunityDetection, and CommunitiesSCD.

#### 5.10.3.8 void CommunityDetection::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:

- CommunityDetection.h

## 5.11 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.*

#### 5.11.1 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

#### 5.11.2 Constructor & Destructor Documentation

##### 5.11.2.1 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. |

#### 5.11.3 Member Function Documentation

##### 5.11.3.1 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.

#### 5.11.3.2 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.

#### 5.11.3.3 sparksee::gdb::Objects* ConnectedComponents::GetNodes ( sparksee::gdb::int64_t *idConnectedComponent* )

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

**Parameters**

| | | |
|---|---|---|
| in | *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.

#### 5.11.3.4 sparksee::gdb::int64_t ConnectedComponents::GetSize ( sparksee::gdb::int64_t *idConnectedComponent* )

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

**Parameters**

| | | |
|---|---|---|
| in | *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.

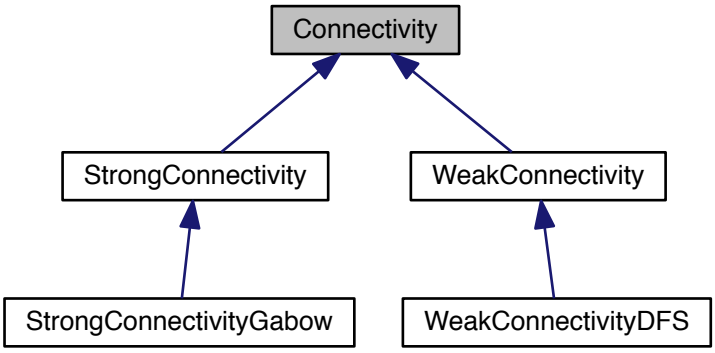
The documentation for this class was generated from the following file:

- ConnectedComponents.h

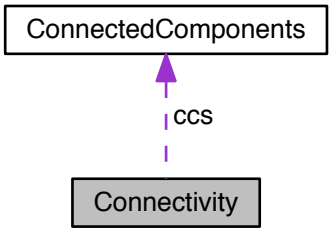
## 5.12 Connectivity Class Reference

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.*

### 5.12.1 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, 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

### 5.12.2 Constructor & Destructor Documentation

#### 5.12.2.1 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 |

### 5.12.3 Member Function Documentation

#### 5.12.3.1 void Connectivity::AddAllEdgeTypes ( sparksee::gdb::EdgesDirection *d* ) [protected]

Allows connectivity through all edge types of the graph.

**Parameters**

| | |
|---|---|
| *d* | [in] Edge direction. |

#### 5.12.3.2 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. |

#### 5.12.3.3 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. |

#### 5.12.3.4 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. |

#### 5.12.3.5 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.

#### 5.12.3.6 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 WeakConnectivityDFS, and StrongConnectivityGabow.

#### 5.12.3.7 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. |

#### 5.12.3.8 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

## 5.13 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 &session, sparksee::gdb::oid_t node)

  *Creates a new instance.*

### Static Public Member Functions

- static sparksee::gdb::Objects * Compute (sparksee::gdb::Session &session, 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.*

#### 5.13.1 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

#### 5.13.2 Constructor & Destructor Documentation

##### 5.13.2.1 Context::Context ( sparksee::gdb::Session & *session*, sparksee::gdb::oid_t *node* )

Creates a new instance.

**Parameters**

| | |
|---|---|
| *session* | [in] Session to get the graph from and perform operation. |
| *node* | [in] Node to start traversal from. |

#### 5.13.3 Member Function Documentation

##### 5.13.3.1 void Context::AddAllEdgeTypes ( sparksee::gdb::EdgesDirection *d* )

Allows for traversing all edge types of the graph.

**Parameters**

| | |
|---|---|
| *d* | [in] Edge direction. |

##### 5.13.3.2 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. |

##### 5.13.3.3 sparksee::gdb::Objects* Context::Compute ( )

Gets the resulting collection of nodes.

**Returns**

The resulting collection of nodes.

#### 5.13.3.4 static sparksee::gdb::Objects* Context::Compute ( sparksee::gdb::Session & *session*, 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**

| | |
|---|---|
| *session* | [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. |

**Returns**

Returns an Objects with the computed context of a node.

#### 5.13.3.5 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. |

#### 5.13.3.6 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. |

#### 5.13.3.7 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. |

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

- Context.h

## 5.14 CSVLoader Class Reference

**Static Public Member Functions**

- static void LoadNodes (sparksee::gdb::Graph &graph, const std::wstring &fileName, const std::wstring &nodeType, const std::wstring &separator=std::wstring(L","), sparksee::gdb::bool_t header=true, const std::vector< sparksee::gdb::int32_t > &columns=std::vector< sparksee::gdb::int32_t >(), const std::vector< std::wstring > &attrNames=std::vector< std::wstring >(), const std::vector< sparksee::gdb::DataType > &dataTypes=std::vector< sparksee::gdb::DataType >(), const std::vector< sparksee::gdb::AttributeKind > &attrKinds=std::vector< sparksee::gdb::AttributeKind >())

  *Loads nodes from a CSV file.*
- static void LoadEdges (sparksee::gdb::Graph &graph, const std::wstring &fileName, const std::wstring &edgeType, const std::wstring &tailNodeType, const std::wstring &headNodeType, sparksee::gdb::int32_t tail=0, sparksee::gdb::int32_t head=1, const std::wstring &separator=std::wstring(L","), sparksee::gdb::bool_t directed=true, sparksee::gdb::bool_t header=true, sparksee::gdb::MissingEndpoint onMissingTail=gdb::IsError, sparksee::gdb::MissingEndpoint onMissingHead=gdb::IsError, const std::vector< sparksee::gdb::int32_t > &columns=std::vector< sparksee::gdb::int32_t >(), const std::vector< std::wstring > &attrNames=std::vector< std::wstring >(), const std::vector< sparksee::gdb::DataType > &dataTypes=std::vector< sparksee::gdb::DataType >(), const std::vector< sparksee::gdb::AttributeKind > &attrKinds=std::vector< sparksee::gdb::AttributeKind >())

  *Loads edges from a CSV file.*

### 5.14.1 Member Function Documentation

#### 5.14.1.1 static void CSVLoader::LoadEdges ( sparksee::gdb::Graph & *graph*, const std::wstring & *fileName*, const std::wstring & *edgeType*, const std::wstring & *tailNodeType*, const std::wstring & *headNodeType*, sparksee::gdb::int32_t *tail* = `0`, sparksee::gdb::int32_t *head* = `1`, const std::wstring & *separator* = `std::wstring(L",")`, sparksee::gdb::bool_t *directed* = `true`, sparksee::gdb::bool_t *header* = `true`, sparksee::gdb::MissingEndpoint *onMissingTail* = gdb::IsError, sparksee::gdb::MissingEndpoint *onMissingHead* = gdb::IsError, const std::vector< sparksee::gdb::int32_t > & *columns* = `std::vector< sparksee::gdb::int32_t >()`, const std::vector< std::wstring > & *attrNames* = `std::vector< std::wstring >()`, const std::vector< sparksee::gdb::DataType > & *dataTypes* = `std::vector< sparksee::gdb::DataType >()`, const std::vector< sparksee::gdb::AttributeKind > & *attrKinds* = `std::vector< sparksee::gdb::AttributeKind >()` ) [static]

Loads edges from a CSV file.

**Parameters**

| | |
|---|---|
| *graph* | [in] The graph to load the edges into |
| *fileName* | [in] The name of the file |
| *nodeType* | [in] The type of the edge to load. If it does not exist, it creates it |
| *tailNodeType* | [in] The type of the tail nodes. If it does not exist and onMissingTail is set to "Create", it creates it. Otherwise, an exception is thrown |
| *headNodeType* | [in] The type of the head nodes. It it does not exist and onMissingHead is set to "Create", it creates it. Otherwise, an exception is thrown |
| *tail* | [in] The tail column index. Default: 0 |
| *head* | [in] The head column index. Default: 1 |
| *separator* | [in] The column separator. Default: "," |
| *directed* | [in] True if this edge is directed or not. False otherwise. Default: True |
| *header* | [in] True if the CSV contains a header. False otherwise. Default: True |
| *onMissingTail* | [in] The policy to follow when a tail is missing. Default: "IsError" |
| *onMissingHead* | [in] The policy to follow when a head is missing. Default: "IsError" |
| *columns* | [in] The list of columns to load. tail and head columns must be in this list if this list is specified. If the list is empty, all columns are loaded. Default: empty |
| *attrNames* | [in] The attribute names. If this list is empty and hasHeader is set to true, the header values are used as attribute names. Default: empty |
| *dataTypes* | [in] The dataTypes of the attributes if this do not already exist. If this list is empty, the method tries to infer the data type from the first non-header row. Default: empty |
| *attrKinds* | [in] The attributeKinds of the attributes if these do not already exist. If this list is empty, the attributeKind of the created attributes are set to Basic. Default: empty |

#### 5.14.1.2 static void CSVLoader::LoadNodes ( sparksee::gdb::Graph & *graph*, const std::wstring & *fileName*, const std::wstring & *nodeType*, const std::wstring & *separator* = `std::wstring(L",")`, sparksee::gdb::bool_t *header* = `true`, const std::vector< sparksee::gdb::int32_t > & *columns* = `std::vector< sparksee::gdb::int32_t >()`, const std::vector< std::wstring > & *attrNames* = `std::vector< std::wstring >()`, const std::vector< sparksee::gdb::DataType > & *dataTypes* = `std::vector< sparksee::gdb::DataType >()`, const std::vector< sparksee::gdb::AttributeKind > & *attrKinds* = `std::vector< sparksee::gdb::AttributeKind >()` ) [static]

Loads nodes from a CSV file.

**Parameters**

| | |
|---|---|
| *graph* | [in] The graph to load the nodes into |
| *fileName* | [in] The name of the file |
| *nodeType* | [in] The type of the node to load. If it does not exist, it creates it |
| *separator* | [in] The separator of the CSV file. Default: "," |
| *header* | [in] True if the CSV contains a header. False otherwise. Default: True |
| *columns* | [in] The list of columns to load. tail and head columns mus be in this list if this list is specified. Default: all columns |
| *attrNames* | [in] The attribute names. If this list is empty and hasHeader is set to true, the header values are used as attribute names. Default: empty |
| *dataTypes* | [in] The dataTypes of the attributes if this do not already exist. Default: empty If this list is empty, the method tries to infer the data type from the first non-header row |
| *attrKinds* | [in] The attributeKinds of the attributes if these do not already exist. Default: empty If this list is empty, the attributeKind of the created attributes are set to Basic. Default: empty |

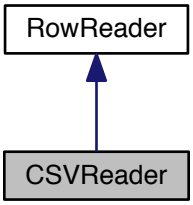
The documentation for this class was generated from the following file:

- CSVLoader.h

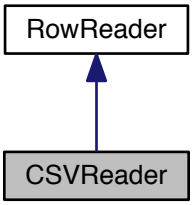
## 5.15 CSVReader Class Reference

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 &quotes) 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 &filePath) throw (sparksee::gdb::IOException)

  *Opens the source file path.*
- sparksee::gdb::bool_t Reset () throw (sparksee::gdb::IOException)

  *Moves the reader to the beginning.*
- virtual 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.*

### 5.15.1 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

### 5.15.2 Member Function Documentation

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

Closes the reader.

**Exceptions**

| | |
|---|---|
| *IOException* | If the close fails. |

Implements RowReader.

#### 5.15.2.2 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.

#### 5.15.2.3 void CSVReader::Open ( const std::wstring & *filePath* ) throw sparksee::gdb::IOException)

Opens the source file path.

File can be optionally compressed in GZIP format.

**Parameters**

| | |
|---|---|
| *filePath* | [in] CSV file path. |

**Exceptions**

| | |
|---|---|
| *IOException* | If bad things happen opening the file. |

#### 5.15.2.4 virtual 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.

#### 5.15.2.5 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.

#### 5.15.2.6 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. |

#### 5.15.2.7 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). |

#### 5.15.2.8 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) |

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

Sets the character used to quote fields.

**Parameters**

| | |
|---|---|
| *quotes* | [in] Quote character. |

#### 5.15.2.10 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. |

#### 5.15.2.11 void CSVReader::SetStartLine ( sparksee::gdb::int32_t *startLine* )

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

**Parameters**

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

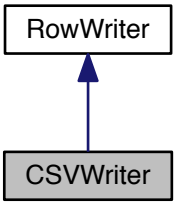
The documentation for this class was generated from the following file:

- CSVReader.h

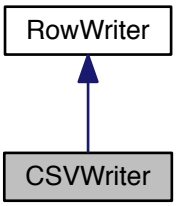
## 5.16 CSVWriter Class Reference

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 &quotes) 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.*

### 5.16.1 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

### 5.16.2 Member Function Documentation

#### 5.16.2.1 void CSVWriter::Close ( ) throw sparksee::gdb::IOException, sparksee::gdb::Error) [virtual]

Closes the writer.

**Exceptions**

| | |
|---|---|
| *IOException* | If the close fails. |

Implements RowWriter.

#### 5.16.2.2 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. |

#### 5.16.2.3 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. |

#### 5.16.2.4 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. |

#### 5.16.2.5 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. |

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

Sets the character used to quote fields.

**Parameters**

| | |
|---|---|
| *quotes* | [in] Quote character. |

#### 5.16.2.7 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. |

#### 5.16.2.8 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.

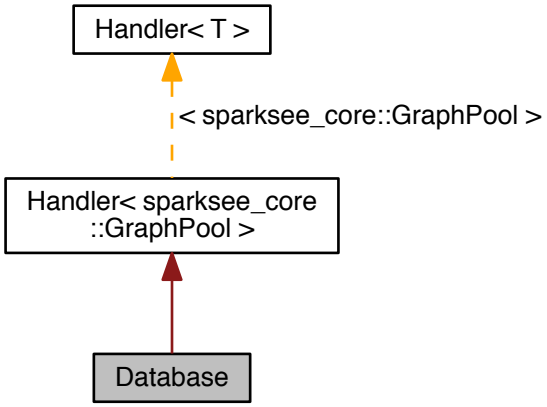
The documentation for this class was generated from the following file:

- CSVWriter.h

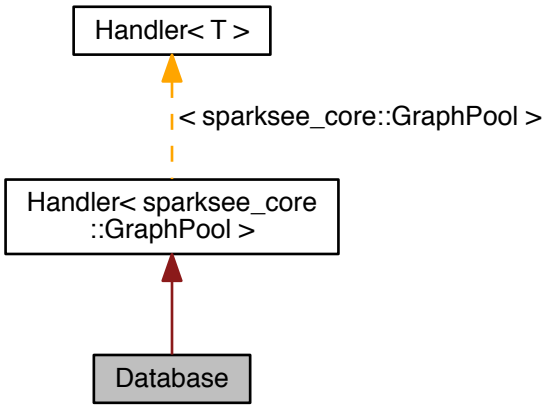
## 5.17 Database Class Reference

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 EnableRollback ()

  *Enables the rollback mechanism.*
- void DisableRollback ()

  *Disables the rollback mechanism.*
- void GetStatistics (DatabaseStatistics &stats)

  *Gets Database statistics.*
- int32_t GetCacheMaxSize ()

  *Gets the cache maximum size (in MB).*
- void SetCacheMaxSize (int32_t megaBytes)

  *Sets the cache maximum size (in MB).*
- void FixCurrentCacheMaxSize ()

  *Sets the cache maximum size to the current cache size in use.*
- void DumpSchema (const std::wstring &filePath)

  *Dump the database schema to the given file using the Sparksee scripting format.*
- void RedoPrecommitted (int64_t txId)

  *Redo a pending precommitted transaction recovered.*

#### Private Member Functions

- sparksee_core::GraphPool * GetHandler ()

  *Gets the handled reference.*
- const sparksee_core::GraphPool * GetHandler () const

  *Gets the handled reference.*
- void SetHandler (sparksee_core::GraphPool *h)

  *Sets the handled reference.*
- void FreeHandler ()

  *Frees (deletes) the handled reference.*
- bool_t IsNull () const

  *Gets if the handler is NULL.*

#### Friends

- class **Sparksee**
- class **Graph**

### 5.17.1 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 SPARKSEEConfig.

#### Author

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

### 5.17.2 Member Function Documentation

#### 5.17.2.1 const std::wstring& Database::GetAlias ( ) const [inline]

Gets the alias of the Database.

**Returns**

The alias of the Database.

#### 5.17.2.2 int32_t Database::GetCacheMaxSize ( )

Gets the cache maximum size (in MB).

**Returns**

Returns the current cache max size.

#### 5.17.2.3 const std::wstring& Database::GetPath ( ) const [inline]

Gets the path of the Database.

**Returns**

The path of the Database.

References END_SPARKSEE_GDB_NAMESPACE.

#### 5.17.2.4 void Database::GetStatistics ( DatabaseStatistics & *stats* )

Gets Database statistics.

**Parameters**

| | |
|---|---|
| *stats* | [out] The DatabaseStatistics instance. |

#### 5.17.2.5 void Database::RedoPrecommitted ( int64_t *txId* )

Redo a pending precommitted transaction recovered.

YOU SHOULD NOT USE THIS METHOD. This method only exists for a specific service.

#### 5.17.2.6 void Database::SetCacheMaxSize ( int32_t *megaBytes* )

Sets the cache maximum size (in MB).

0 means unlimited which is all the physical memory of the computer minus a small margin.

**Parameters**

| | |
|---|---|
| *megaBytes* | [in] The new cache max size. |

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

- Database.h

## 5.18 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**

### 5.18.1 Detailed Description

Database statistics.

**Author**

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

### 5.18.2 Member Function Documentation

#### 5.18.2.1 int64_t DatabaseStatistics::GetCache ( ) const [inline]

Gets cache size in KBytes.

**Returns**

Cache size in KBytes.

#### 5.18.2.2 int64_t DatabaseStatistics::GetData ( ) const [inline]

Gets database size in KBytes.

**Returns**

Database size in KBytes.

#### 5.18.2.3 int64_t DatabaseStatistics::GetRead ( ) const [inline]

Gets total read data in KBytes.

**Returns**

Total read data in KBytes.

#### 5.18.2.4 int64_t DatabaseStatistics::GetSessions ( ) const [inline]

Gets the number of sessions.

**Returns**

The number of sessions.

#### 5.18.2.5 int64_t DatabaseStatistics::GetTemp ( ) const [inline]

Gets temporary storage file size in KBytes.

**Returns**

Temporary storage file size in KBytes.

#### 5.18.2.6 int64_t DatabaseStatistics::GetWrite ( ) const [inline]

Gets total written data in KBytes.

**Returns**

Total read written in KBytes.

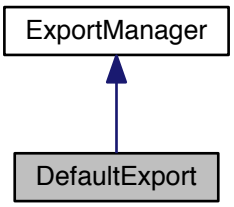
The documentation for this class was generated from the following file:

- Database.h

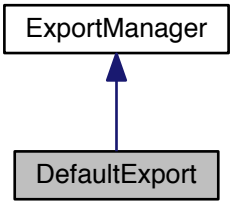
## 5.19 DefaultExport Class Reference

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.*

#### 5.19.1 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

#### 5.19.2 Member Function Documentation

##### 5.19.2.1 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.

**Parameters**

| | |
|---|---|
| *type* | [in] The type to enable. |

**Returns**

TRUE.

Implements ExportManager.

##### 5.19.2.2 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.

**Parameters**

| | |
|---|---|
| *edge* | [in] An edge. |
| *edgeExport* | [out] The EdgeExport that will store the information. |

**Returns**

TRUE.

Implements ExportManager.

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

Default implementation of the ExportManager class method GetEdgeType.

This sets de default EdgeExport values.

**Parameters**

| | |
|---|---|
| *type* | [in] An edge type. |
| *edgeExport* | [out] The EdgeExport that will store the information. |

**Returns**

TRUE.

Implements ExportManager.

#### 5.19.2.4 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.

**Parameters**

| | |
|---|---|
| *graphExport* | [out] The GraphExport that will store the information. |

**Returns**

TRUE.

Implements ExportManager.

#### 5.19.2.5 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.

**Parameters**

| | |
|---|---|
| *node* | [in] A node. |
| *nodeExport* | [out] The NodeExport that will store the information. |

**Returns**

TRUE.

Implements ExportManager.

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

Default implementation of the ExportManager class method GetNodeType.

This sets de default NodeExport values.

**Parameters**

| | |
|---|---|
| *type* | [in] A node type. |
| *nodeExport* | [out] The NodeExport that will store the information. |

**Returns**

TRUE.

Implements ExportManager.

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

- Export.h

## 5.20 DisjointCommunities Class Reference

DisjointCommunities class.

**Public Member Functions**

- DisjointCommunities (sparksee::gdb::Session &session, const std::wstring &materializedattribute)

  *Creates a new instance of DisjointCommunities.*
- virtual ~DisjointCommunities ()

  *Destructor.*
- sparksee::gdb::int64_t GetCommunity (sparksee::gdb::oid_t idNode)

  *Returns the disjoint community where the given node belongs to.*
- sparksee::gdb::int64_t GetCount ()

  *Returns the number of communities found in the graph.*
- sparksee::gdb::Objects * GetNodes (sparksee::gdb::int64_t idCommunity)

  *Returns the collection of nodes contained in the given community.*
- sparksee::gdb::int64_t GetSize (sparksee::gdb::int64_t idCommunity)

  *Returns the number of nodes contained in the given community.*

### 5.20.1 Detailed Description

DisjointCommunities class.

This class contains the results processed on a DisjointCommunityDetection algorithm.

These results contain information related to the communities found. We must consider that each community 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 communities found.

When executing any implementation of the DisjointCommunityDetection, it is possible to indicate whether the results of the execution must be stored persistently using the class DisjointCommunityDetection 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

### 5.20.2 Constructor & Destructor Documentation

#### 5.20.2.1 DisjointCommunities::DisjointCommunities ( sparksee::gdb::Session & *session*, const std::wstring & *materializedattribute* )

Creates a new instance of DisjointCommunities.

This constructor method can only be called when a previous execution of any implementation of the DisjointCommunityDetection 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 DisjointCommunityDetection execution see the documentation of the DisjointCommunityDetection#SetMaterializedAttribute method.

#### Parameters

| | |
|---|---|
| *session* | [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 communities found in the graph. |

### 5.20.3 Member Function Documentation

#### 5.20.3.1 sparksee::gdb::int64_t DisjointCommunities::GetCommunity ( sparksee::gdb::oid_t *idNode* )

Returns the disjoint community where the given node belongs to.

#### Parameters

| | |
|---|---|
| *idNode* | [in] The node identifier for which the disjoint community identifier where it belongs will be returned. |

**Returns**

The disjoint community identifier where the given node identifier belongs to.

#### 5.20.3.2 sparksee::gdb::int64_t DisjointCommunities::GetCount ( )

Returns the number of communities found in the graph.

**Returns**

The number of communities found in the graph.

#### 5.20.3.3 sparksee::gdb::Objects* DisjointCommunities::GetNodes ( sparksee::gdb::int64_t *idCommunity* )

Returns the collection of nodes contained in the given community.

**Parameters**

| | | |
|---|---|---|
| in | *idCommunity* | The community for which the collection of nodes contained in it will be returned. |

**Returns**

The collection of node identifiers contained in the given community.

#### 5.20.3.4 sparksee::gdb::int64_t DisjointCommunities::GetSize ( sparksee::gdb::int64_t *idCommunity* )

Returns the number of nodes contained in the given community.

**Parameters**

| | | |
|---|---|---|
| in | *idCommunity* | The community for which the number of nodes contained in it will be returned. |

**Returns**

The number of nodes contained in the given community.

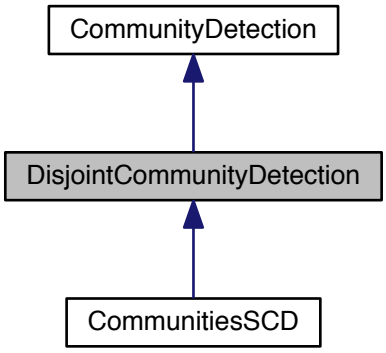
The documentation for this class was generated from the following file:

- DisjointCommunities.h

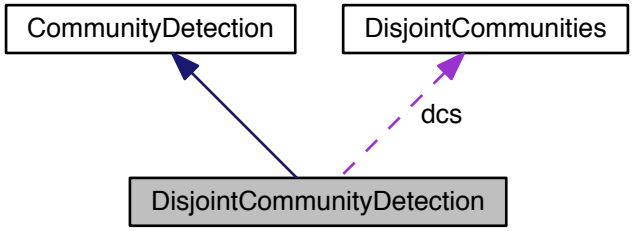
## 5.21 DisjointCommunityDetection Class Reference

DisjointCommunityDetection class.

Inheritance diagram for DisjointCommunityDetection:




Collaboration diagram for DisjointCommunityDetection:




**Public Member Functions**

- virtual ~DisjointCommunityDetection ()

  *Destructor.*
- virtual void AddEdgeType (sparksee::gdb::type_t type)

  *Allows connectivity through edges of the given type.*
- virtual void AddAllEdgeTypes ()

  *Allows connectivity through all edge types of the graph.*
- DisjointCommunities * GetCommunities ()

  *Returns the results generated by the execution of the algorithm.*
- virtual void Run ()=0

  *Runs the algorithm in order to find the communities.*
- 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 disjoint communities found while executing this algorithm.*

- virtual void AddNodeType (sparksee::gdb::type_t type)

  *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.*
- virtual void IncludeNodes (sparksee::gdb::Objects &nodes)

  *Set additional nodes that can be used.*
- virtual void IncludeEdges (sparksee::gdb::Objects &edges)

  *Set additional edges that can be used.*

#### 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

- DisjointCommunityDetection (sparksee::gdb::Session &s)

  *Creates a new instance of DisjointCommunityDetection.*
- void SetCommunity (sparksee::gdb::oid_t idNode)

  *Assigns the current community to the given node.*
- void AssertNotCommunityAttribute (const std::wstring &attributeName)

  *Check that the given attribute name is not already in use.*
- void CreateGlobalPersistentAttribute (const std::wstring &attributeName)

  *Set a new persistent global attribute to store the disjoint communities information.*
- void CreateGlobalTransientAttribute ()

  *Set a new temporary global attribute to store the disjoint communities information.*
- void RemoveGlobalAttribute ()

  *Removes the global attribute where the disjoint communities information is stored.*
- void AddEdgeType (sparksee::gdb::type_t type, sparksee::gdb::EdgesDirection direction)

  *Allows connectivity through edges of the given type.*
- void AddAllEdgeTypes (sparksee::gdb::EdgesDirection direction)

  *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 SetNodesNotVisited ()

  *Set all the selected nodes in nodesNotVisited.*
- void AssertComputed ()

  *Check that the communities 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.*
- sparksee::gdb::bool_t IsNodeTypeAllowed (sparksee::gdb::oid_t nodeId)

  *Check if the given node has an allowed type.*
- sparksee::gdb::bool_t IsEdgeTypeAllowed (sparksee::gdb::oid_t edgeId)

  *Check if the given edge has an allowed type.*
- sparksee::gdb::bool_t IsNodeAllowed (sparksee::gdb::oid_t nodeId)

  *Check if the given node is allowed (by type or by explicit inclusion).*
- sparksee::gdb::bool_t IsEdgeAllowed (sparksee::gdb::oid_t edgeId)

  *Check if the given edge is allowed (by type or by explicit inclusion).*
- 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.*
- sparksee::gdb::bool_t IsNodeIncluded (sparksee::gdb::oid_t node)

  *Check if the given node is EXPLICITLY allowed using the include set.*
- sparksee::gdb::bool_t IsEdgeIncluded (sparksee::gdb::oid_t edge)

  *Check if the given edge is EXPLICITLY allowed using the include set.*

#### Protected Attributes

- sparksee::gdb::attr_t attrCommunity

  *common attribute where the connected component information is stored.*
- std::wstring attrCommunityName

  *name of the common attribute where the connected component information is stored.*
- sparksee::gdb::int64_t actualCommunity

  *Current community identifier.*
- sparksee::gdb::bool_t matResults

  *Materialized results.*
- DisjointCommunities * dcs

  *The calculated communities information.*
- 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::Objects * nodesNotVisited

  *Identifiers of the nodes not visited.*
- 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.*
- sparksee::gdb::Objects * includedNodes

  *The set of explicitly included nodes.*
- sparksee::gdb::Objects * includedEdges

  *The set of explicitly included edges.*

#### 5.21.1 Detailed Description

DisjointCommunityDetection class.

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

#### 5.21.2 Constructor & Destructor Documentation

##### 5.21.2.1 DisjointCommunityDetection::DisjointCommunityDetection ( sparksee::gdb::Session & *s* ) [protected]

Creates a new instance of DisjointCommunityDetection.

**Parameters**

| | |
|---|---|
| *s* | [in] Session to get the graph from and calculate the communities |

#### 5.21.3 Member Function Documentation

##### 5.21.3.1 virtual void DisjointCommunityDetection::AddAllEdgeTypes ( ) [virtual]

Allows connectivity through all edge types of the graph.

The edges can be used in Any direction.

##### 5.21.3.2 void CommunityDetection::AddAllEdgeTypes ( sparksee::gdb::EdgesDirection *direction* ) [protected], [inherited]

Allows connectivity through all edge types of the graph.

**Parameters**

| | |
|---|---|
| *d* | [in] Edge direction. |

##### 5.21.3.3 virtual void DisjointCommunityDetection::AddEdgeType ( sparksee::gdb::type_t *type* ) [virtual]

Allows connectivity through edges of the given type.

The edges can be used in Any direction.

**Parameters**

| | |
|---|---|
| *type* | [in] Edge type. |

#### 5.21.3.4 void CommunityDetection::AddEdgeType ( sparksee::gdb::type_t *type*, sparksee::gdb::EdgesDirection *direction* ) [protected], [inherited]

Allows connectivity through edges of the given type.

**Parameters**

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

#### 5.21.3.5 virtual void CommunityDetection::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. |

#### 5.21.3.6 virtual void CommunityDetection::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. |

#### 5.21.3.7 DisjointCommunities* DisjointCommunityDetection::GetCommunities ( )

Returns the results generated by the execution of the algorithm.

These results contain information related to the disjoint communities 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 DisjointCommunities which contain information related to the disjoint communities found.

#### 5.21.3.8 virtual void CommunityDetection::IncludeEdges ( sparksee::gdb::Objects & *edges* ) [virtual], [inherited]

Set additional edges that can be used.

This will replace any previously specified set of include edges. Using this optional method adds valid edges to the edges of any edge type explicitly set as a valid type. Should only be used to include specific small sets of edges because it's less efficient than just using an edge type. For any edge to be used, both nodes must be also valid.

**Parameters**

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

#### 5.21.3.9 virtual void CommunityDetection::IncludeNodes ( sparksee::gdb::Objects & *nodes* ) [virtual], [inherited]

Set additional nodes that can be used.

This will replace any previously specified set of include nodes. Using this optional method adds valid nodes to the nodes of any node type explicitly set as a valid type. Should only be used to include specific small sets of nodes because it's less efficient than just using a node type.

**Parameters**

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

#### 5.21.3.10 virtual void DisjointCommunityDetection::Run ( ) [pure virtual]

Runs the algorithm in order to find the communities.

This method can be called only once.

Implements CommunityDetection.

Implemented in CommunitiesSCD.

#### 5.21.3.11 void DisjointCommunityDetection::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 disjoint communities 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 DisjointCommunities 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 disjoint communities 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. |

#### 5.21.3.12 void CommunityDetection::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:

- DisjointCommunityDetection.h

## 5.22 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**

### 5.22.1 Detailed Description

Edge data class.

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

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

#### Author

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

### 5.22.2 Member Function Documentation

#### 5.22.2.1 oid_t EdgeData::GetEdge ( ) const [inline]

Gets the edge identifier.

#### Returns

The Sparksee edge identifier.

#### 5.22.2.2 oid_t EdgeData::GetHead ( ) const [inline]

Gets the head of the edge.

**Returns**

The Sparksee edge identifier of the head of the edge.

#### 5.22.2.3 oid_t EdgeData::GetTail ( ) const [inline]

Gets the tail of the edge.

**Returns**

The Sparksee edge identifier of the tail of the edge.

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

- Graph_data.h

## 5.23 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 &label)
  *Sets the edge label.*
- bool_t AsDirected () const
  *Gets if the edge should be managed as directed.*
- void SetAsDirected (bool_t directed)
  *Sets if the edge should be managed as directed.*
- ColorRGB GetColorRGB () const
  *Gets the edge color.*
- void SetColorRGB (ColorRGB color)
  *Sets the edge color.*
- ColorRGB GetLabelColorRGB () const
  *Gets the edge label color.*
- void SetLabelColorRGB (ColorRGB color)
  *Sets the edge label color.*
- int32_t GetWidth () const
  *Gets the edge width.*
- void SetWidth (int32_t width)
  *Sets the edge width.*
- int32_t GetFontSize () const
  *Gets the edge label font size.*
- void SetFontSize (int32_t size)
  *Sets the edge label font size.*

### 5.23.1 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 (OxD3D3D3, Light gray).

Label color: 0 (Ox000000, Black).

Width: 5px.

Font size: 10.

**Author**

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

### 5.23.2 Member Function Documentation

#### 5.23.2.1 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.

#### 5.23.2.2 ColorRGB EdgeExport::GetColorRGB ( ) const [inline]

Gets the edge color.

**Returns**

The edge color.

#### 5.23.2.3 int32_t EdgeExport::GetFontSize ( ) const [inline]

Gets the edge label font size.

**Returns**

The edge label font size.

#### 5.23.2.4 const std::wstring& EdgeExport::GetLabel ( ) const [inline]

Gets the edge label.

**Returns**

The edge label.

#### 5.23.2.5 ColorRGB EdgeExport::GetLabelColorRGB ( ) const [inline]

Gets the edge label color.

**Returns**

The edge label color.

#### 5.23.2.6 int32_t EdgeExport::GetWidth ( ) const [inline]

Gets the edge width.

**Returns**

The edge width.

#### 5.23.2.7 void EdgeExport::SetAsDirected ( bool_t *directed* ) [inline]

Sets if the edge should be managed as directed.

**Parameters**

| | |
|---|---|
| *directed* | [in] If TRUE, use as directed, otherwise use as undirected. |

#### 5.23.2.8 void EdgeExport::SetColorRGB ( ColorRGB *color* ) [inline]

Sets the edge color.

**Parameters**

| | |
|---|---|
| *color* | [in] The edge color. |

#### 5.23.2.9 void EdgeExport::SetFontSize ( int32_t *size* ) [inline]

Sets the edge label font size.

**Parameters**

| | |
|---|---|
| *size* | [in] The edge label font size. |

#### 5.23.2.10 void EdgeExport::SetLabel ( const std::wstring & *label* ) [inline]

Sets the edge label.

**Parameters**

| | |
|---|---|
| *label* | [in] The edge label. |

#### 5.23.2.11 void EdgeExport::SetLabelColorRGB ( ColorRGB *color* ) [inline]

Sets the edge label color.

**Parameters**

| | |
|---|---|
| *color* | [in] The edge label color. |

#### 5.23.2.12 void EdgeExport::SetWidth ( int32_t *width* ) [inline]

Sets the edge width.

**Parameters**

| | |
|---|---|
| *width* | [in] The edge width. |

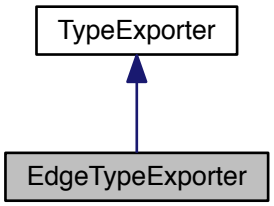
The documentation for this class was generated from the following file:

- Export.h

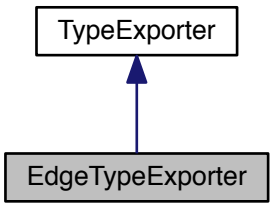
## 5.24 EdgeTypeExporter Class Reference

EdgeTypeExporter class.

Inheritance diagram for EdgeTypeExporter:



Collaboration diagram for EdgeTypeExporter:




### Public Member Functions

- EdgeTypeExporter ()

  *Creates a new instance.*
- EdgeTypeExporter (RowWriter &rowWriter, sparksee::gdb::Graph &graph, sparksee::gdb::type_t type, 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 &graph)

  *Sets the graph that will be exported.*
- void SetType (sparksee::gdb::type_t type)

  *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.*

### 5.24.1 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

### 5.24.2 Constructor & Destructor Documentation

#### 5.24.2.1 EdgeTypeExporter::EdgeTypeExporter ( RowWriter & *rowWriter*, sparksee::gdb::Graph & *graph*, sparksee::gdb::type_t *type*, 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

| | |
|---|---|
| *rowWriter* | [in] Output RowWriter. |
| *graph* | [in] Graph. |
| *type* | [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. |

References TypeExporter::SetHeadAttribute(), TypeExporter::SetHeadPosition(), TypeExporter::SetTailAttribute(), and TypeExporter::SetTailPosition().

### 5.24.3 Member Function Documentation

#### 5.24.3.1 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.

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

Notifies progress to all registered listeners.

**Parameters**

| | |
|---|---|
| *ev* | [in] Progress event to be notified. |

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

Registers a new listener.

**Parameters**

| | |
|---|---|
| *tel* | [in] TypeExporterListener to be registered. |

#### 5.24.3.4 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. |

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

Sets the list of Attributes.

**Parameters**

| | |
|---|---|
| *attrs* | [in] Attribute identifiers to be exported |

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

Sets the frequency of listener notification.

**Parameters**

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

#### 5.24.3.7 void TypeExporter::SetGraph ( sparksee::gdb::Graph & *graph* ) [inherited]

Sets the graph that will be exported.

**Parameters**

| | |
|---|---|
| *graph* | [in] Graph. |

#### 5.24.3.8 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 |

References TypeExporter::SetHeadAttribute().

#### 5.24.3.9 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. |

#### 5.24.3.10 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 |

References TypeExporter::SetHeadPosition().

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

Sets the output data destination.

**Parameters**

| | |
|---|---|
| *rw* | [in] Input RowWriter. |

#### 5.24.3.12 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 |

References TypeExporter::SetTailAttribute().

#### 5.24.3.13 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 |

References END_SPARKSEE_IO_NAMESPACE, and TypeExporter::SetTailPosition().

#### 5.24.3.14 void TypeExporter::SetType ( sparksee::gdb::type_t *type* ) [inherited]

Sets the type to be exported.

**Parameters**

| | |
|---|---|
| *type* | [in] Type identifier. |

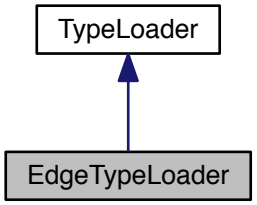
The documentation for this class was generated from the following file:

- EdgeTypeExporter.h

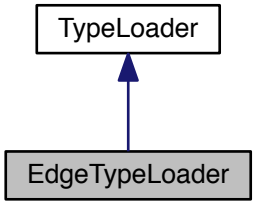
## 5.25 EdgeTypeLoader Class Reference

EdgeTypeLoader class.

Inheritance diagram for EdgeTypeLoader:



Collaboration diagram for EdgeTypeLoader:




### Public Member Functions

- EdgeTypeLoader ()

  *Creates a new instance.*
- EdgeTypeLoader (RowReader &rowReader, sparksee::gdb::Graph &graph, sparksee::gdb::type_t type, 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, sparksee::gdb::MissingEndpoint headMEP, sparksee::gdb::MissingEndpoint tailMEP)

  *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 SetTailMEP (sparksee::gdb::MissingEndpoint mep)

  *Sets the flag to create missing tail nodes when loading nodes or edges.*
- void SetHeadMEP (sparksee::gdb::MissingEndpoint mep)

  *Sets the flag to create missing head nodes when loading nodes or edges.*
- 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 &graph)

  *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 type)

  *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 &timestampFormat)

  *Sets a specific timestamp format.*
- void SetFrequency (sparksee::gdb::int32_t freq)

  *Sets the frequency of listener notification.*

#### Protected Types

#### Protected Member Functions

- void Run (Mode ph, sparksee::gdb::int32_t par) throw (sparksee::gdb::IOException, sparksee::gdb::Error)

  *Runs load process.*
- sparksee::gdb::bool_t CanRun ()

  *Checks that all the required settings are ready to run.*
- void NotifyListeners (const TypeLoaderEvent &ev)

  *Notifies progress to all registered listeners.*

### 5.25.1 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

### 5.25.2 Member Enumeration Documentation

#### 5.25.2.1 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.

### 5.25.3 Constructor & Destructor Documentation

#### 5.25.3.1 EdgeTypeLoader::EdgeTypeLoader ( RowReader & *rowReader,* sparksee::gdb::Graph & *graph,* sparksee::gdb::type_t *type,* 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,* sparksee::gdb::MissingEndpoint *headMEP,* sparksee::gdb::MissingEndpoint *tailMEP* ) [inline]

Creates a new instance.

**Parameters**

| | |
|---|---|
| *rowReader* | [in] Input RowReader. |
| *graph* | [in] Graph. |
| *type* | [in] Type identifier. |
| *attrs* | [in] Attribute identifiers to be loaded. |
| *attrsPos* | [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. |
| *tailMEP* | [in] The MissingEndpoint policy for the tail nodes |
| *headMEP* | [in] The MissingEndpoint policy for the head nodes |

References TypeLoader::SetHeadAttribute(), TypeLoader::SetHeadMEP(), TypeLoader::SetHeadPosition(), TypeLoader::SetTailAttribute(), TypeLoader::SetTailMEP(), and TypeLoader::SetTailPosition().

### 5.25.4 Member Function Documentation

#### 5.25.4.1 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.

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

Notifies progress to all registered listeners.

**Parameters**

| | |
|---|---|
| *ev* | [in] Progress event to be notified. |

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

Registers a new listener.

**Parameters**

| | | |
|---|---|---|
| in | *tel* | TypeLoaderListener to be registered. |

#### 5.25.4.4 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. |

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

Sets the list of attribute positions.

**Parameters**

| | |
|---|---|
| *attrsPos* | [in] Attribute positions (column index >=0). |

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

Sets the list of Attributes.

**Parameters**

| | |
|---|---|
| *attrs* | [in] Attribute identifiers to be loaded |

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

Sets the frequency of listener notification.

**Parameters**

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

#### 5.25.4.8 void TypeLoader::SetGraph ( sparksee::gdb::Graph & *graph* ) [inherited]

Sets the graph where the data will be loaded.

**Parameters**

| | |
|---|---|
| *graph* | [in] Graph. |

#### 5.25.4.9 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 |

References TypeLoader::SetHeadAttribute().

#### 5.25.4.10 void EdgeTypeLoader::SetHeadMEP ( sparksee::gdb::MissingEndpoint *mep* ) [inline]

Sets the flag to create missing head nodes when loading nodes or edges.

**Parameters**

| | |
|---|---|
| *flag* | True if the nodes need to be created. False otherwise |

References END_SPARKSEE_IO_NAMESPACE, and TypeLoader::SetHeadMEP().

#### 5.25.4.11 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 |

References TypeLoader::SetHeadPosition().

#### 5.25.4.12 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. |

#### 5.25.4.13 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.

**Parameters**

| | |
|---|---|
| *path* | [in] The path to the error log file. |

**Exceptions**

| | |
|---|---|
| *IOException* | If bad things happen opening the file. |

#### 5.25.4.14 void TypeLoader::SetLogOff ( ) [inherited]

Truns 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.

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

Sets the input data source.

**Parameters**

| | |
|---|---|
| *rr* | [in] Input RowReader. |

#### 5.25.4.16 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 |

References TypeLoader::SetTailAttribute().

#### 5.25.4.17 void EdgeTypeLoader::SetTailMEP ( sparksee::gdb::MissingEndpoint *mep* ) [inline]

Sets the flag to create missing tail nodes when loading nodes or edges.

**Parameters**

| | |
|---|---|
| *flag* | True if the nodes need to be created. False otherwise |

References TypeLoader::SetTailMEP().

#### 5.25.4.18 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 |

References TypeLoader::SetTailPosition().

#### 5.25.4.19 void TypeLoader::SetTimestampFormat ( const std::wstring & *timestampFormat* ) [inherited]

Sets a specific timestamp format.

**Parameters**

| | |
|---|---|
| *timestampFormat* | [in] A string with the timestamp format definition. |

#### 5.25.4.20 void TypeLoader::SetType ( sparksee::gdb::type_t *type* ) [inherited]

Sets the type to be loaded.

**Parameters**

| | |
|---|---|
| *type* | [in] Type identifier. |

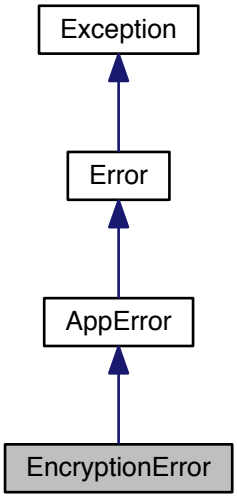
The documentation for this class was generated from the following file:

- EdgeTypeLoader.h

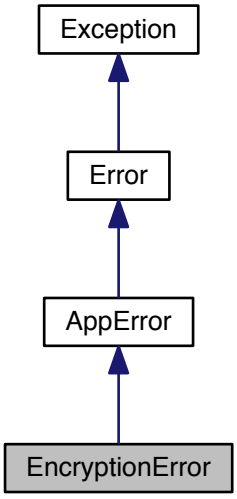
## 5.26 EncryptionError Class Reference

Wrong encryption arguments error class.

Inheritance diagram for EncryptionError:




Collaboration diagram for EncryptionError:




**Public Member Functions**

- EncryptionError ()

  *Creates a new instance.*

- EncryptionError (const std::string &mess)

  *Creates a new instance.*
- virtual ~EncryptionError ()

  *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 void ThrowError (int32_t coreErrorCode)

  *Throws a new Error instance from a sparksee_core error code.*

#### Protected Attributes

- std::string message

  *Message of the exception.*

### 5.26.1 Detailed Description

Wrong encryption arguments error class.

#### Author

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

### 5.26.2 Constructor & Destructor Documentation

#### 5.26.2.1 EncryptionError::EncryptionError ( const std::string & *mess* )

Creates a new instance.

#### Parameters

| | |
|---|---|
| *mess* | [in] Message of the exception. |

### 5.26.3 Member Function Documentation

#### 5.26.3.1 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.

#### 5.26.3.2 void Exception::SetMessage ( const std::string & *mess* ) [inherited]

Sets the message of the exception.

**Parameters**

| | |
|---|---|
| *mess* | [in] Message. |

#### 5.26.3.3 static void Error::ThrowError ( int32_t *coreErrorCode* ) [static], [inherited]

Throws 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 will throw an Error instance of an specific Error subclass instance.

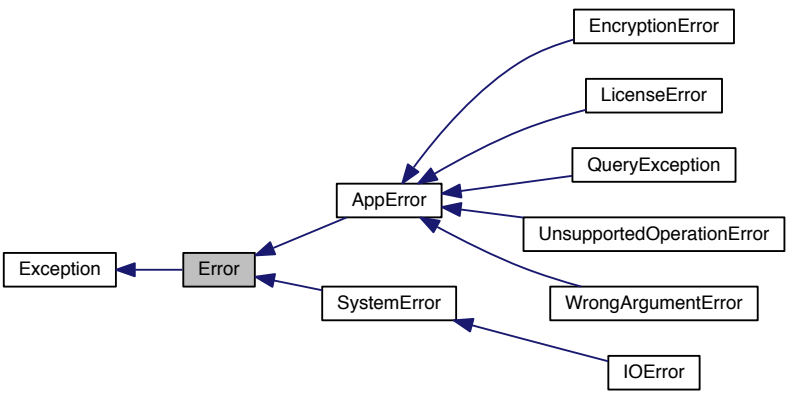
The documentation for this class was generated from the following file:

- Exception.h

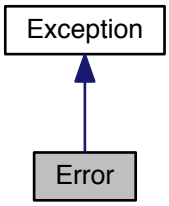
## 5.27 Error Class Reference

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 void ThrowError (int32_t coreErrorCode)
  *Throws a new Error instance from a sparksee_core error code.*

#### Protected Attributes

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

### 5.27.1 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

### 5.27.2 Constructor & Destructor Documentation

#### 5.27.2.1 Error::Error ( const std::string & *mess* )

Creates a new instance.

**Parameters**

| | |
|---|---|
| *mess* | [in] Message of the exception. |

### 5.27.3 Member Function Documentation

#### 5.27.3.1 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.

#### 5.27.3.2 void Exception::SetMessage ( const std::string & *mess* ) [inherited]

Sets the message of the exception.

**Parameters**

| | |
|---|---|
| *mess* | [in] Message. |

#### 5.27.3.3 static void Error::ThrowError ( int32_t *coreErrorCode* ) [static]

Throws 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 will throw an Error instance of an specific Error subclass instance.

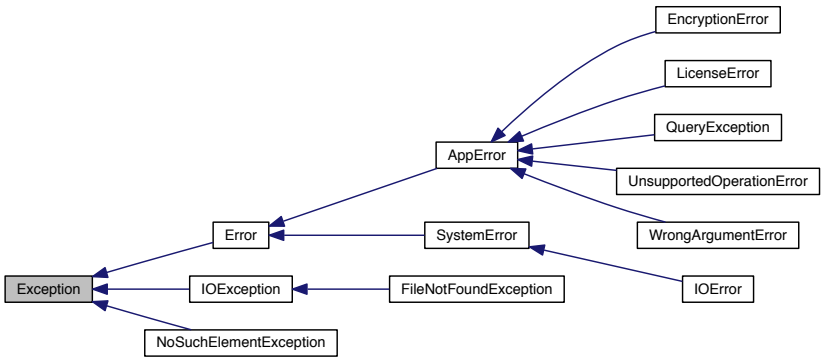
The documentation for this class was generated from the following file:

- Exception.h

## 5.28 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.*

### 5.28.1 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

### 5.28.2 Constructor & Destructor Documentation

#### 5.28.2.1 Exception::Exception ( const std::string & *mess* )

Creates a new instance.

**Parameters**

| | |
|---|---|
| *mess* | [in] Message of the exception. |

#### 5.28.3 Member Function Documentation

##### 5.28.3.1 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.

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

Sets the message of the exception.

**Parameters**

| | |
|---|---|
| *mess* | [in] Message. |

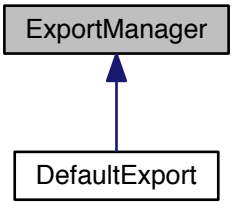
The documentation for this class was generated from the following file:

- Exception.h

### 5.29 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.*

### 5.29.1 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 fortmats. Nowadays, available formats are defined in the ExportType enum.

#### Author

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

### 5.29.2 Member Function Documentation

#### 5.29.2.1 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.

#### 5.29.2.2 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.

#### 5.29.2.3 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.

#### 5.29.2.4 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.

#### 5.29.2.5 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.

#### 5.29.2.6 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.

#### 5.29.2.7 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.

#### 5.29.2.8 virtual void ExportManager::Release ( ) [pure virtual]

Ends the export process.

It is called once after the export process.

Implemented in DefaultExport.

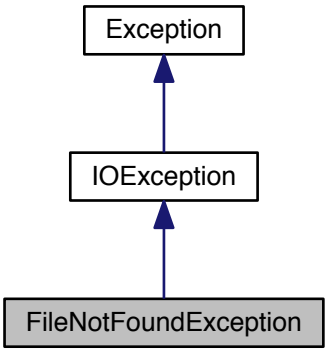
The documentation for this class was generated from the following file:

- Export.h

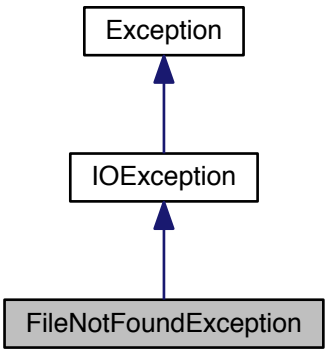
## 5.30 FileNotFoundException Class Reference

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.*

### 5.30.1 Detailed Description

File not found exception class.

#### Author

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

### 5.30.2 Constructor & Destructor Documentation

#### 5.30.2.1 FileNotFoundException::FileNotFoundException ( 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. |

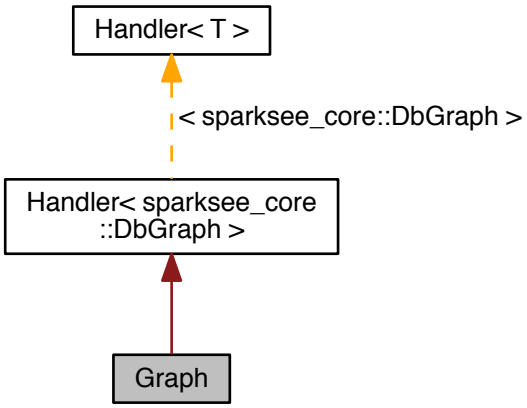
The documentation for this class was generated from the following file:

- Exception.h

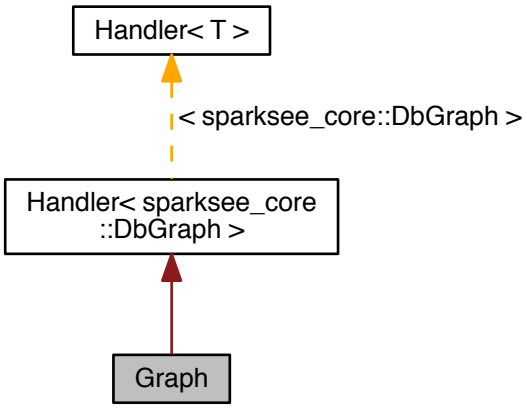
## 5.31 Graph Class Reference

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.*
- void IndexNeighbors (type_t edgeType, bool_t neighbors)

  *Creates or destroys the neighbors index of an 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 &default↩ Value)

  *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.*
- attr_t NewArrayAttribute (type_t type, const std::wstring &name, DataType dt, int32_t size)

  *Creates a new array attribute.*
- attr_t NewSessionArrayAttribute (type_t type, DataType dt, int32_t size)

  *Creates a new Session array attribute.*
- void SetAttributeDefaultValue (attr_t attr, Value &value)

  *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 &value)

  *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.*
- ValueArray * GetArrayAttribute (oid_t oid, attr_t attr)

  *Gets the ValueArray for the given array attribute and OID or NULL if it does not exist.*
- ValueArray * SetArrayAttribute (oid_t oid, attr_t attr, const Value &value)

  *Sets all the elements of the ValueArray for the given array attribute and OID and returns it.*
- void SetArrayAttributeVoid (oid_t oid, attr_t attr, const Value &value)

  *Sets all the elements of the ValueArray for the given array attribute and OID.*
- void SetArrayAttribute (attr_t attr, const Value &value)

  *Sets all the values of the array of the given array attribute for all the objects of the types the attribute applies to.*
- void SetAttribute (oid_t oid, attr_t attr, Value &value)

  *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.*
- void SetAttribute (attr_t attr, Value &value)

  *Sets the value of the given attribute for all the objects of the types the attribute applies to.*
- 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 &value)

  *Finds one object having the given Value for the given attribute.*
- oid_t FindOrCreateObject (attr_t attr, Value &value)

  *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 &value)

  *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 &value, 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.*
- KeyValues * TopK (attr_t attribute, Order order, int32_t k)

  *Gets a KeyValues iterator as a result of the TopK operation.*
- KeyValues * TopK (attr_t attribute, Condition operation, const Value &lower, Order order, int32_t k)

  *Gets a KeyValues iterator as a result of the TopK operation.*
- KeyValues * TopK (attr_t attribute, Condition operation, const Value &lower, const Value &higher, Order order, int32_t k)

  *Gets a KeyValues iterator as a result of the TopK operation.*
- KeyValues * TopK (attr_t attribute, Order order, int32_t k, Objects *restriction)

  *Gets a KeyValues iterator as a result of the TopK operation.*
- KeyValues * TopK (attr_t attribute, Condition operation, const Value &lower, Order order, int32_t k, Objects *restriction)

  *Gets a KeyValues iterator as a result of the TopK operation.*
- KeyValues * TopK (attr_t attribute, Condition operation, const Value &lower, const Value &higher, Order order, int32_t k, Objects *restriction)

  *Gets a KeyValues iterator as a result of the TopK operation.*
- 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 *em) throw (sparksee::gdb::IO↩ Exception)

  *Exports the Graph.*
- void Backup (const std::wstring &file) throw (sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)

  *Dumps all the data to a backup file.*
- void EncryptedBackup (const std::wstring &file, const std::wstring &keyInHex, const std::wstring &ivInHex) throw (sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)

  *Dumps all the data to a backup file.*

#### Private Member Functions

- sparksee_core::DbGraph * GetHandler ()

  *Gets the handled reference.*
- const sparksee_core::DbGraph * GetHandler () const

  *Gets the handled reference.*
- void SetHandler (sparksee_core::DbGraph *h)

  *Sets the handled reference.*
- void FreeHandler ()

  *Frees (deletes) the handled reference.*
- bool_t IsNull () const

  *Gets if the handler is NULL.*

#### Friends

- class **Session**
- class **Values**
- class **ValuesIterator**

### 5.31.1 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

### 5.31.2 Member Function Documentation

#### 5.31.2.1 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 methods.

**Parameters**

| | |
|---|---|
| *file* | [in] Output backup file path. |

**Exceptions**

| | |
|---|---|
| *FileNotFoundException* | If the given file cannot be created. |

#### 5.31.2.2 int64_t Graph::CountEdges ( )

Gets the number of edges.

**Returns**

The number of edges.

#### 5.31.2.3 int64_t Graph::CountNodes ( )

Gets the number of nodes.

**Returns**

The number of nodes.

#### 5.31.2.4 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.

#### 5.31.2.5 void Graph::Drop ( oid_t *oid* )

Drops the given OID.

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

**Parameters**

| | |
|---|---|
| *oid* | [in] Sparksee OID to be removed. |

#### 5.31.2.6 void Graph::Drop ( Objects * *objs* )

Drops all the OIDs from the given collection.

See Drop method with the single OID parameter. This performs the same operation for each object in the given set.

**Parameters**

| | |
|---|---|
| *objs* | [in] Objects collection with the OIDs to be removed. |

#### 5.31.2.7 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. |

#### 5.31.2.8 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. |

#### 5.31.2.9 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.

#### 5.31.2.10 void Graph::EncryptedBackup ( const std::wstring & *file*, const std::wstring & *keyInHex*, const std::wstring & *ivInHex* ) throw sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)

Dumps all the data to a backup file.

See the Sparksee class RestoreEncryptedBackup methods.

**Parameters**

| | |
|---|---|
| *file* | [in] Output backup file path. |
| *keyInHex* | [In] The AES encryption Key as an hexadecimal string (8, 16 or 32 bytes). |
| *ivInHex* | [In] The AES Initialization Vector as an hexadecimal string (16 bytes). |

**Exceptions**

| | |
|---|---|
| *FileNotFoundException* | If the given file cannot be created. |

#### 5.31.2.11 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.

#### 5.31.2.12 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.

#### 5.31.2.13 void Graph::Export ( const std::wstring & *file*, ExportType *type*, ExportManager * *em* ) throw sparksee::gdb::IOException)

Exports the Graph.

**Parameters**

| | |
|---|---|
| *file* | [in] Output file. |
| *type* | [in] Export type. |
| *em* | [in] Defines how to do the export for each graph object. |

#### 5.31.2.14 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.

#### 5.31.2.15 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.

#### 5.31.2.16 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.

#### 5.31.2.17 TypeList* Graph::FindEdgeTypes ( )

Gets all existing Sparksee edge type identifiers.

**Returns**

Sparksee edge type identifier list.

#### 5.31.2.18 TypeList* Graph::FindNodeTypes ( )

Gets all existing Sparksee node type identifiers.

**Returns**

Sparksee node type identifier list.

#### 5.31.2.19 oid_t Graph::FindObject ( attr_t *attr*, Value & *value* )

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. |
| *value* | [in] Value. |

**Returns**

Sparksee OID or the Objects InvalidOID.

#### 5.31.2.20 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.

#### 5.31.2.21 oid_t Graph::FindOrCreateObject ( attr_t *attr*, Value & *value* )

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. |
| *value* | [in] Value. |

**Returns**

Sparksee OID or the Objects InvalidOID.

#### 5.31.2.22 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.

#### 5.31.2.23 TypeList* Graph::FindTypes ( )

Gets all existing Sparksee node and edge type identifiers.

**Returns**

Sparksee node and edge type identifier list.

#### 5.31.2.24 ValueArray* Graph::GetArrayAttribute ( oid_t *oid*, attr_t *attr* )

Gets the ValueArray for the given array attribute and OID or NULL if it does not exist.

**Parameters**

| | |
|---|---|
| *oid* | [in] Sparksee OID. |
| *attr* | [in] Sparksee array attribute identifier. |

**Returns**

A ValueArray. This returns a ValueArray, a NULL or throws an exception.

#### 5.31.2.25 void Graph::GetAttribute ( oid_t *oid*, attr_t *attr*, Value & *value* )

Gets the Value for the given attribute and OID.

**Parameters**

| | |
|---|---|
| *oid* | [in] Sparksee OID. |
| *attr* | [in] Sparksee attribute identifier. |
| *value* | [in\|out] Value for the given attribute and for the given OID. |

#### 5.31.2.26 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.

#### 5.31.2.27 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.

#### 5.31.2.28 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.

#### 5.31.2.29 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.

#### 5.31.2.30 AttributeStatistics* Graph::GetAttributeStatistics ( attr_t *attr*, bool_t *basic* )

Gets statistics from the given attribute.

The statistics can only be obtained from Indexed or Unique attributes.

**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 instace.

#### 5.31.2.31 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.

#### 5.31.2.32 EdgeData* Graph::GetEdgeData ( oid_t *edge* )

Gets information about an edge.

**Parameters**

| | |
|---|---|
| *edge* | [in] Sparksee edge identifier. |

**Returns**

An EdgeData instance.

#### 5.31.2.33 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.

#### 5.31.2.34 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.

#### 5.31.2.35 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.

#### 5.31.2.36 Values* Graph::GetValues ( attr_t *attr* )

Gets the Value collection for the given attribute.

**Parameters**

| | |
|---|---|
| *attr* | [in] Sparksee attribute identifier. |

**Returns**

Returns a Values object.

#### 5.31.2.37 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.

#### 5.31.2.38 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. |

#### 5.31.2.39 void Graph::IndexNeighbors ( type_t *edgeType*, bool_t *neighbors* )

Creates or destroys the neighbors index of an edge type.

**Parameters**

| | |
|---|---|
| *edgeType* | [in] Sparksee Edge type identifier. |
| *neighbors* | [in] If TRUE, it indexes the neighbor nodes, otherwise it removes the index. |

#### 5.31.2.40 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.

#### 5.31.2.41 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.

#### 5.31.2.42 attr_t Graph::NewArrayAttribute ( type_t *type*, const std::wstring & *name*, DataType *dt*, int32_t *size* )

Creates a new array attribute.

**Parameters**

| | |
|---|---|
| *type* | [in] Sparksee node or edge type identifier. |
| *name* | [in] Unique name for the new attribute. |
| *dt* | [in] Base Data type for the new array attribute elements. |
| *size* | [in] The array size. |

**Returns**

Unique Sparksee attribute identifier.

#### 5.31.2.43 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.

#### 5.31.2.44 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.

#### 5.31.2.45 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.

#### 5.31.2.46 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.

#### 5.31.2.47 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 a undirected edge type. |
| *neighbors* | [in] If TRUE, this indexes neighbor nodes, otherwise not. |

**Returns**

Unique Sparksee type identifier.

#### 5.31.2.48 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.

#### 5.31.2.49 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.

#### 5.31.2.50 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.

#### 5.31.2.51 attr_t Graph::NewSessionArrayAttribute ( type_t *type,* DataType *dt,* int32_t *size* )

Creates a new Session array 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] Base Data type for the new array attribute elements. |
| *size* | [in] The array size. |

**Returns**

Unique Sparksee attribute identifier.

#### 5.31.2.52 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.

#### 5.31.2.53 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.

#### 5.31.2.54 void Graph::RemoveAttribute ( attr_t *attr* )

Removes the given attribute.

**Parameters**

| | |
|---|---|
| *attr* | [in] Sparksee attribute identifier. |

#### 5.31.2.55 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. |

#### 5.31.2.56 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. |

#### 5.31.2.57 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. |

#### 5.31.2.58 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. |

#### 5.31.2.59 Objects* Graph::Select ( type_t *type* )

Selects all OIDs belonging to the given type.

**Parameters**

| | |
|---|---|
| *type* | [in] Sparksee type identifier. |

**Returns**

Objects instance.

#### 5.31.2.60 Objects* Graph::Select ( attr_t *attr*, Condition *cond*, const Value & *value* )

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

**Parameters**

| | |
|---|---|
| *attr* | [in] Sparksee attribute identifier. |
| *cond* | [in] Condition to be satisfied. |
| *value* | [in] Value to be satisfied. |

**Returns**

Objects instance.

#### 5.31.2.61 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.

#### 5.31.2.62 Objects* Graph::Select ( attr_t *attr*, Condition *cond*, const Value & *value*, 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. |
| *value* | [in] Value to be satisfied. |
| *restriction* | [in] Objects to limit the select in this set of objects. |

**Returns**

Objects instance.

#### 5.31.2.63 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.

#### 5.31.2.64 ValueArray* Graph::SetArrayAttribute ( oid_t *oid*, attr_t *attr*, const Value & *value* )

Sets all the elements of the ValueArray for the given array attribute and OID and returns it.

Initializing the array with one of these SetArrayAttribute methods is the only way to create the array. But once created it can be updated using the ValueArray. And you can get it again with the GetArrayAttribute operation.

**Parameters**

| | |
|---|---|
| *oid* | [in] Sparksee OID. |
| *attr* | [in] Sparksee array attribute identifier. |
| *value* | [in] Value for all the array elements of the given attribute and OID. |

**Returns**

A ValueArray. This returns a ValueArray, a NULL or throws an exception.

#### 5.31.2.65 void Graph::SetArrayAttribute ( attr_t *attr*, const Value & *value* )

Sets all the values of the array of the given array attribute for all the objects of the types the attribute applies to.

**Parameters**

| | |
|---|---|
| *attr* | [in] Sparksee array attribute identifier. |
| *value* | [in] Value for all the array elements of the arrays. |

#### 5.31.2.66 void Graph::SetArrayAttributeVoid ( oid_t *oid*, attr_t *attr*, const Value & *value* )

Sets all the elements of the ValueArray for the given array attribute and OID.

Initializing the array with one of these SetArrayAttribute methods is the only way to create the array. But once created it can be updated using the ValueArray which can be obtained using the GetArrayAttribute operation.

**Parameters**

| | |
|---|---|
| *oid* | [in] Sparksee OID. |
| *attr* | [in] Sparksee array attribute identifier. |
| *value* | [in] Value for all the array elements of the given attribute and OID. |

#### 5.31.2.67 void Graph::SetAttribute ( oid_t *oid*, attr_t *attr*, Value & *value* )

Sets the Value for the given attribute and OID.

**Parameters**

| | |
|---|---|
| *oid* | [in] Sparksee OID. |
| *attr* | [in] Sparksee attribute identifier. |
| *value* | [in] Value for the given attribute and for the given OID. |

#### 5.31.2.68 void Graph::SetAttribute ( attr_t *attr*, Value & *value* )

Sets the value of the given attribute for all the objects of the types the attribute applies to.

Does not work for TEXT attribute types

**Parameters**

| | |
|---|---|
| *tx* | The Transaction this operation belongs to |
| *attribute* | The attribute to initialize |
| *value* | The value to initialize the attribute to |

#### 5.31.2.69 void Graph::SetAttributeDefaultValue ( attr_t *attr*, Value & *value* )

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. |
| *value* | [in] The default value to use for this attribute. |

#### 5.31.2.70 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. |

#### 5.31.2.71 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.

#### 5.31.2.72 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. |

#### 5.31.2.73 KeyValues* Graph::TopK ( attr_t *attribute*, Order *order*, int32_t *k* )

Gets a KeyValues iterator as a result of the TopK operation.

**Parameters**

| | |
|---|---|
| *attribute* | The attribute to perform the TopK on |
| *order* | The ordering semantincs of the top-k |
| *k* | The size of the TopK |

**Returns**

A KeyValues instance

#### 5.31.2.74 KeyValues* Graph::TopK ( attr_t *attribute*, Condition *operation*, const Value & *lower*, Order *order*, int32_t *k* )

Gets a KeyValues iterator as a result of the TopK operation.

**Parameters**

| | |
|---|---|
| *attribute* | The attribute to perform the TopK on |
| *operation* | The operation to perform in the top k |
| *lower* | The lower bound Value to be satisfied |
| *order* | The ordering semantincs of the top-k |
| *k* | The size of the TopK |

**Returns**

A KeyValues instance

#### 5.31.2.75 KeyValues* Graph::TopK ( attr_t *attribute*, Condition *operation*, const Value & *lower*, const Value & *higher*, Order *order*, int32_t *k* )

Gets a KeyValues iterator as a result of the TopK operation.

**Parameters**

| | |
|---|---|
| *attribute* | The attribute to perform the TopK on |
| *operation* | The operation to perform in the top k |
| *lower* | The lower bound Value to be satisfied |
| *higher* | The upper bound Value to be satisfied |
| *order* | The ordering semantincs of the top-k |
| *k* | The size of the TopK |

**Returns**

A KeyValues instance

#### 5.31.2.76 KeyValues* Graph::TopK ( attr_t *attribute*, Order *order*, int32_t *k*, Objects * *restriction* )

Gets a KeyValues iterator as a result of the TopK operation.

**Parameters**

| | |
|---|---|
| *attribute* | The attribute to perform the TopK on |
| *order* | The ordering semantincs of the top-k |
| *k* | The size of the TopK |
| *restricton* | Objects to limit the topk to this set of objects |

**Returns**

A KeyValues instance

#### 5.31.2.77 KeyValues* Graph::TopK ( attr_t *attribute*, Condition *operation*, const Value & *lower*, Order *order*, int32_t *k*, Objects * *restriction* )

Gets a KeyValues iterator as a result of the TopK operation.

**Parameters**

| | |
|---|---|
| *attribute* | The attribute to perform the TopK on |
| *operation* | The operation to perform in the top k |
| *lower* | The lower bound Value to be satisfied |
| *order* | The ordering semantincs of the top-k |
| *k* | The size of the TopK |
| *restricton* | Objects to limit the topk to this set of objects |

**Returns**

A KeyValues instance

#### 5.31.2.78 KeyValues* Graph::TopK ( attr_t *attribute*, Condition *operation*, const Value & *lower*, const Value & *higher*, Order *order*, int32_t *k*, Objects * *restriction* )

Gets a KeyValues iterator as a result of the TopK operation.

**Parameters**

| | |
|---|---|
| *attribute* | The attribute to perform the TopK on |
| *operation* | The operation to perform in the top k |
| *lower* | The lower bound Value to be satisfied |
| *higher* | The upper bound Value to be satisfied |
| *order* | The ordering semantincs of the top-k |
| *k* | The size of the TopK |
| *restricton* | Objects to limit the topk to this set of objects |

**Returns**

A KeyValues instance

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

- Graph.h

## 5.32 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 &label)

  *Sets the graph label.*

### 5.32.1 Detailed Description

Stores the graph exporting values.

#### Author

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

### 5.32.2 Member Function Documentation

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

Gets the graph label.

#### Returns

The graph label.

#### 5.32.2.2 void GraphExport::SetLabel ( const std::wstring & *label* ) [inline]

Sets the graph label.

#### Parameters

| | |
|---|---|
| *label* | [in] The graph label. |

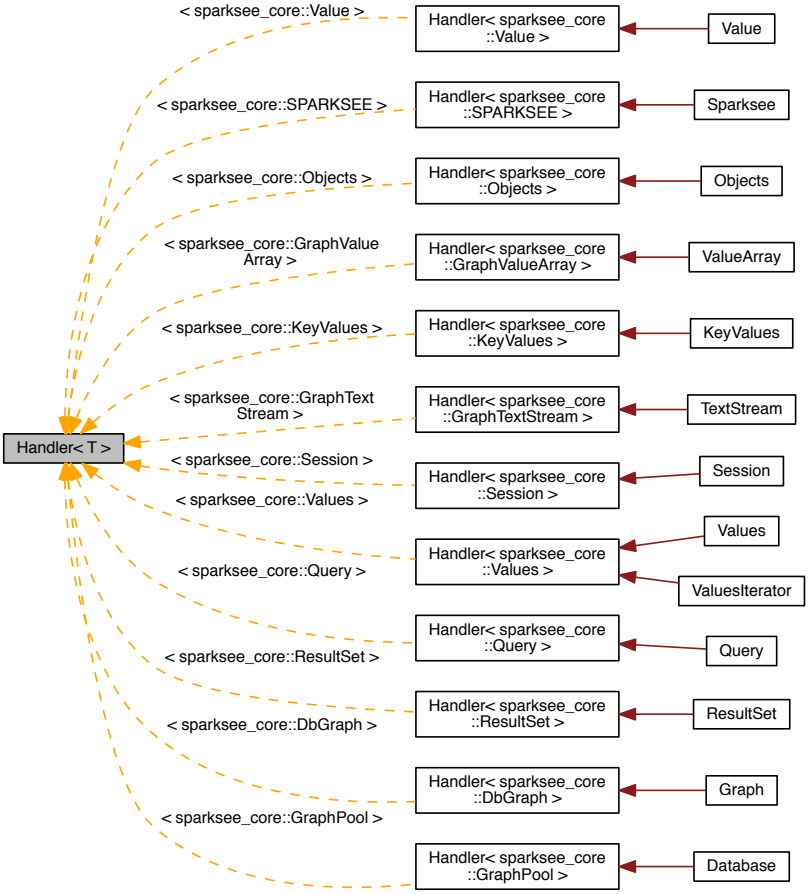
The documentation for this class was generated from the following file:

- Export.h

## 5.33 Handler< T > Class Template Reference

Handles a reference.

Inheritance 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.*

### 5.33.1 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

### 5.33.2 Constructor & Destructor Documentation

#### 5.33.2.1 template<typename T> Handler< T >::Handler ( T * *h* ) [inline]

Creates a new instance with the given reference.

**Parameters**

| | |
|---|---|
| *h* | [in] Reference to be handled. |

#### 5.33.2.2 template<typename T> virtual Handler< T >::~Handler ( ) [inline], [virtual]

Destructor.

Frees the handled reference.

### 5.33.3 Member Function Documentation

#### 5.33.3.1 template<typename T> T* Handler< T >::GetHandler ( ) [inline]

Gets the handled reference.

**Returns**

The handled reference.

Referenced by Value::SetVoid().

#### 5.33.3.2 template<typename T> const T* Handler< T >::GetHandler ( ) const [inline]

Gets the handled reference.

**Returns**

The handled reference.

#### 5.33.3.3 template<typename T> bool_t Handler< T >::IsNull ( ) const [inline], [protected]

Gets if the handler is NULL.

**Returns**

TRUE if the handler is NULL, FALSE otherwise.

#### 5.33.3.4 template<typename T> void Handler< T >::SetHandler ( T * *h* ) [inline], [protected]

Sets the handled reference.

**Parameters**

| | |
|---|---|
| *h* | [in] The handled reference. |

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

- Handler.h

## 5.34 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.*

### 5.34.1 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

#### 5.34.2 Constructor & Destructor Documentation

##### 5.34.2.1 Int32List::Int32List ( )

Constructor.

This creates an empty list.

##### 5.34.2.2 Int32List::Int32List ( const std::vector< int32_t > & *v* )

Constructor.

**Parameters**

| | |
|---|---|
| *v* | [in] Vector. |

#### 5.34.3 Member Function Documentation

##### 5.34.3.1 void Int32List::Add ( int32_t *value* ) [inline]

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

**Parameters**

| | |
|---|---|
| *value* | [in] The integer. |

##### 5.34.3.2 int32_t Int32List::Count ( ) const [inline]

Number of elements in the list.

**Returns**

Number of elements in the list.

##### 5.34.3.3 Int32ListIterator* Int32List::Iterator ( )

Gets a new Int32ListIterator.

**Returns**

Int32ListIterator instance.

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

- Graph_data.h

## 5.35 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**

### 5.35.1 Detailed Description

Int32List iterator class.

Iterator to traverse all the integer into a Int32List instance.

#### Author

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

### 5.35.2 Member Function Documentation

#### 5.35.2.1 bool_t Int32ListIterator::HasNext ( ) [inline]

Gets if there are more elements.

#### Returns

TRUE if there are more elements, FALSE otherwise.

References END_SPARKSEE_GDB_NAMESPACE.

#### 5.35.2.2 int32_t Int32ListIterator::Next ( ) [inline]

Moves to the next element.

#### Returns

The next element.

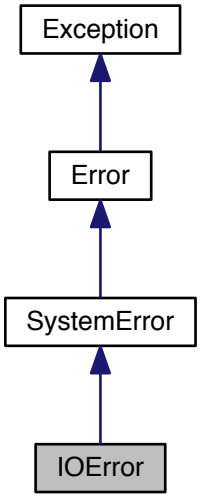
The documentation for this class was generated from the following file:

- Graph_data.h

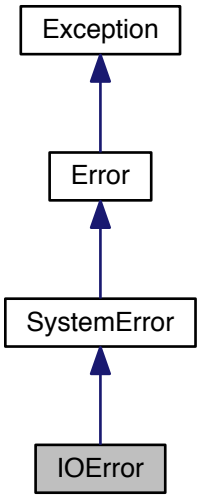
## 5.36 IOError Class Reference

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 void ThrowError (int32_t coreErrorCode)
  *Throws a new Error instance from a sparksee_core error code.*

#### Protected Attributes

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

### 5.36.1 Detailed Description

IO error class.

#### Author

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

### 5.36.2 Constructor & Destructor Documentation

#### 5.36.2.1 IOError::IOError ( const std::string & *mess* )

Creates a new instance.

#### Parameters

| | |
|---|---|
| *mess* | [in] Message of the exception. |

### 5.36.3 Member Function Documentation

#### 5.36.3.1 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.

#### 5.36.3.2 void Exception::SetMessage ( const std::string & *mess* ) [inherited]

Sets the message of the exception.

**Parameters**

| | |
|---|---|
| *mess* | [in] Message. |

#### 5.36.3.3 static void Error::ThrowError ( int32_t *coreErrorCode* ) [static],[inherited]

Throws 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 will throw an Error instance of an specific Error subclass instance.

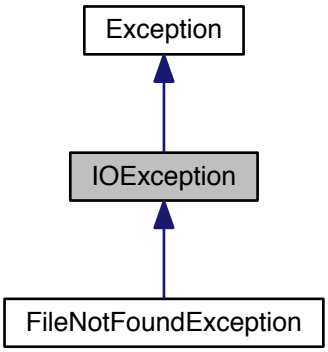
The documentation for this class was generated from the following file:

- Exception.h

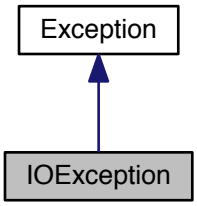
## 5.37 IOException Class Reference

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.*

### 5.37.1 Detailed Description

IO exception class.

#### Author

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

### 5.37.2 Constructor & Destructor Documentation

#### 5.37.2.1 IOException::IOException ( const std::string & *mess* )

Creates a new instance.

**Parameters**

| | |
|---|---|
| *mess* | [in] Message of the exception. |

### 5.37.3 Member Function Documentation

#### 5.37.3.1 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.

#### 5.37.3.2 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

## 5.38 KeyValue Class Reference

**Friends**

- class **KeyValues**

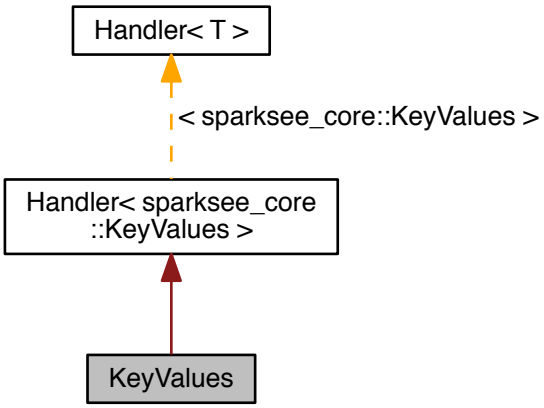
The documentation for this class was generated from the following file:

- KeyValues.h

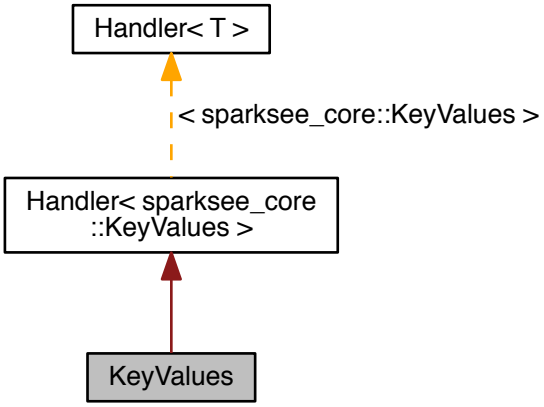
## 5.39 KeyValues Class Reference

Value set class.

Inheritance diagram for KeyValues:




Collaboration diagram for KeyValues:




**Public Member Functions**

- virtual ~KeyValues ()

  *Destructor.*
- bool_t HasNext ()

  *Checks if the KeyValues has more KeyValue pairs.*
- KeyValue * Next ()

  *Gets the next KeyValue pair.*
- void Next (KeyValue &kv)

  *Gets the next KeyValue pair.*

#### Private Member Functions

- sparksee_core::KeyValues * GetHandler ()
  *Gets the handled reference.*
- const sparksee_core::KeyValues * GetHandler () const
  *Gets the handled reference.*
- void SetHandler (sparksee_core::KeyValues *h)
  *Sets the handled reference.*
- void FreeHandler ()
  *Frees (deletes) the handled reference.*
- bool_t IsNull () const
  *Gets if the handler is NULL.*

#### Friends

- class **Graph**

### 5.39.1 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

### 5.39.2 Member Function Documentation

#### 5.39.2.1 bool_t KeyValues::HasNext ( )

Checks if the KeyValues has more KeyValue pairs.

#### Returns

Returns ture if there are more KeyValue pairs

#### 5.39.2.2 KeyValue* KeyValues::Next ( )

Gets the next KeyValue pair.

#### Returns

Returns the next KeyValue pair

#### 5.39.2.3 void KeyValues::Next ( KeyValue & *kv* )

Gets the next KeyValue pair.

#### Parameters

| | |
|---|---|
| *kv* | Returns the next KeyValue pair |

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

- KeyValues.h

## 5.40 KOpt Class Reference

KOpt class.

#### Public Member Functions

- KOpt (sparksee::gdb::Session &session)

  *Creates a new instance.*
- KOpt (sparksee::gdb::Session &session, sparksee::gdb::OIDList &tour)

  *Creates a new instance.*
- virtual ~KOpt ()

  *Destructor.*
- void AddNodeType (sparksee::gdb::type_t type) throw (sparksee::gdb::Error)

  *Allows for traversing nodes of the given type.*
- void AddAllNodeTypes ()

  *Allows for traversing all node types of the graph.*
- void AddEdgeType (sparksee::gdb::type_t type, sparksee::gdb::EdgesDirection dir) throw (sparksee::gdb::↩ Error)

  *Allows for traversing edges of the given type.*
- void AddAllEdgeTypes (sparksee::gdb::EdgesDirection dir)

  *Allows for traversing all edge types of the graph.*
- void SetEdgeWeightAttributeType (sparksee::gdb::attr_t attr) throw (sparksee::gdb::Error)

  *Sets the attribute to use as edge weight.*
- double GetCurrentCost () const

  *Returns tour cost.*
- sparksee::gdb::OIDList * GetCurrentTour ()

  *Returns tour as a list of nodes.*
- void SetCurrentTour (sparksee::gdb::OIDList &tour)

  *Sets current tour as a list of nodes.*
- void SetMaxIterations (sparksee::gdb::int64_t maxIterations)

  *Sets maximum number of iterations.*
- void SetTimeLimit (sparksee::gdb::int64_t maxTime)

  *Limits execution time.*
- void RunTwoOpt ()

  *Runs 2-Opt local search.*
- void RunThreeOpt ()

  *Runs 3-Opt local search.*

#### 5.40.1 Detailed Description

KOpt class.

Implements the 2-Opt and 3-Opt algorithms

**Author**

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

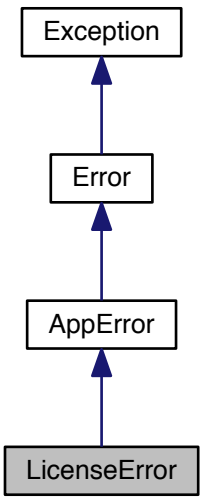
The documentation for this class was generated from the following file:

- KOpt.h

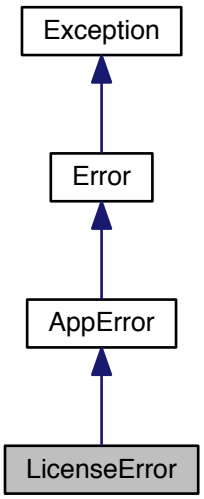
## 5.41 LicenseError Class Reference

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 void ThrowError (int32_t coreErrorCode)

  *Throws a new Error instance from a sparksee_core error code.*

#### Protected Attributes

- std::string message

  *Message of the exception.*

#### 5.41.1 Detailed Description

License error class.

**Author**

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

#### 5.41.2 Constructor & Destructor Documentation

##### 5.41.2.1 LicenseError::LicenseError ( const std::string & *mess* )

Creates a new instance.

**Parameters**

| | |
|---|---|
| *mess* | [in] Message of the exception. |

#### 5.41.3 Member Function Documentation

##### 5.41.3.1 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.

##### 5.41.3.2 void Exception::SetMessage ( const std::string & *mess* ) [inherited]

Sets the message of the exception.

**Parameters**

| | |
|---|---|
| *mess* | [in] Message. |

##### 5.41.3.3 static void Error::ThrowError ( int32_t *coreErrorCode* ) [static], [inherited]

Throws 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 will throw an Error instance of an specific Error subclass instance.

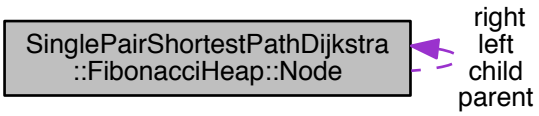
The documentation for this class was generated from the following file:

- Exception.h

## 5.42 SinglePairShortestPathDijkstra::FibonacciHeap::Node Struct Reference

A FibonacciHeap node structure.

Collaboration diagram for SinglePairShortestPathDijkstra::FibonacciHeap::Node:




### 5.42.1 Detailed Description

A FibonacciHeap node structure.

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

- SinglePairShortestPathDijkstra.h

## 5.43 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 &label)
  *Sets the node label.*
- NodeShape GetShape () const
  *Gets the node shape.*
- void SetShape (NodeShape shape)
  *Sets the node shape.*
- ColorRGB GetColorRGB () const
  *Gets the node color.*
- void SetColorRGB (ColorRGB color)
  *Sets the node color.*
- ColorRGB GetLabelColorRGB () const
  *Gets the node label color.*
- void SetLabelColorRGB (ColorRGB color)
  *Sets the node label color.*
- int32_t GetHeight () const
  *Gets the node height.*
- void SetHeight (int32_t height)
  *Sets the node height.*
- int32_t GetWidth () const
  *Gets the node width.*
- void SetWidth (int32_t width)
  *Gets the node width.*
- bool_t IsFit () const
  *Gets whether the node size is fitted to the label or not.*
- void SetFit (bool_t fit)
  *Sets the node fit property.*
- int32_t GetFontSize () const
  *Gets the node label font size.*
- void SetFontSize (int32_t size)
  *Sets the node label font size.*

### 5.43.1 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 (Ox000000, Black).

Height: 25px.

Width: 25px.

Fit: TRUE.

Font size: 10.

**Author**

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

#### 5.43.2 Member Function Documentation

##### 5.43.2.1 ColorRGB NodeExport::GetColorRGB ( ) const [inline]

Gets the node color.

**Returns**

The node color.

##### 5.43.2.2 int32_t NodeExport::GetFontSize ( ) const [inline]

Gets the node label font size.

**Returns**

The node label font size.

##### 5.43.2.3 int32_t NodeExport::GetHeight ( ) const [inline]

Gets the node height.

**Returns**

The node height in pixels.

##### 5.43.2.4 const std::wstring& NodeExport::GetLabel ( ) const [inline]

Gets the node label.

**Returns**

The node label.

#### 5.43.2.5 ColorRGB NodeExport::GetLabelColorRGB ( ) const [inline]

Gets the node label color.

**Returns**

The node label color.

#### 5.43.2.6 NodeShape NodeExport::GetShape ( ) const [inline]

Gets the node shape.

**Returns**

The node shape.

#### 5.43.2.7 int32_t NodeExport::GetWidth ( ) const [inline]

Gets the node width.

**Returns**

The node width in pixels.

#### 5.43.2.8 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'.

#### 5.43.2.9 void NodeExport::SetColorRGB ( ColorRGB *color* ) [inline]

Sets the node color.

**Parameters**

| | |
|---|---|
| *color* | The node color. |

#### 5.43.2.10 void NodeExport::SetFit ( bool_t *fit* ) [inline]

Sets the node fit property.

**Parameters**

| | |
|---|---|
| *fit* | [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'. |

#### 5.43.2.11 void NodeExport::SetFontSize ( int32_t *size* ) [inline]

Sets the node label font size.

**Parameters**

| | |
|---|---|
| *size* | [in] The node label font size. |

#### 5.43.2.12 void NodeExport::SetHeight ( int32_t *height* ) [inline]

Sets the node height.

**Parameters**

| | |
|---|---|
| *height* | [in] The node height in pixels. |

#### 5.43.2.13 void NodeExport::SetLabel ( const std::wstring & *label* ) [inline]

Sets the node label.

**Parameters**

| | |
|---|---|
| *label* | [in] The node label. |

#### 5.43.2.14 void NodeExport::SetLabelColorRGB ( ColorRGB *color* ) [inline]

Sets the node label color.

**Parameters**

| | |
|---|---|
| *color* | [in] The node label color. |

#### 5.43.2.15 void NodeExport::SetShape ( NodeShape *shape* ) [inline]

Sets the node shape.

**Parameters**

| | |
|---|---|
| *shape* | [in] The node shape. |

#### 5.43.2.16 void NodeExport::SetWidth ( int32_t *width* ) [inline]

Gets the node width.

**Parameters**

| | |
|---|---|
| *width* | The node width in pixels. |

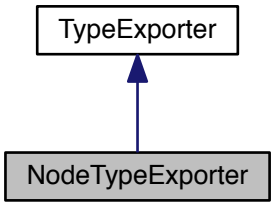
The documentation for this class was generated from the following file:

- Export.h

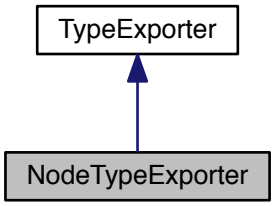
## 5.44 NodeTypeExporter Class Reference

NodeTypeExporter class.

Inheritance diagram for NodeTypeExporter:




Collaboration diagram for NodeTypeExporter:




### Public Member Functions

- NodeTypeExporter ()

  *Creates a new instance.*
- NodeTypeExporter (RowWriter &rowWriter, sparksee::gdb::Graph &graph, sparksee::gdb::type_t type, 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 &graph)

  *Sets the graph that will be exported.*
- void SetType (sparksee::gdb::type_t type)

  *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.*

### 5.44.1 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

### 5.44.2 Constructor & Destructor Documentation

#### 5.44.2.1 NodeTypeExporter::NodeTypeExporter ( RowWriter & *rowWriter,* sparksee::gdb::Graph & *graph,* sparksee::gdb::type_t *type,* sparksee::gdb::AttributeList & *attrs* ) [inline]

Creates a new instance.

**Parameters**

| | |
|---|---|
| *rowWriter* | [in] Output RowWriter. |
| *graph* | [in] Graph. |
| *type* | [in] Type identifier. |
| *attrs* | [in] Attribute identifiers to be exported. |

### 5.44.3 Member Function Documentation

#### 5.44.3.1 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.

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

Notifies progress to all registered listeners.

**Parameters**

| | |
|---|---|
| *ev* | [in] Progress event to be notified. |

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

Registers a new listener.

**Parameters**

| | |
|---|---|
| *tel* | [in] TypeExporterListener to be registered. |

#### 5.44.3.4 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. |

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

Sets the list of Attributes.

**Parameters**

| | |
|---|---|
| *attrs* | [in] Attribute identifiers to be exported |

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

Sets the frequency of listener notification.

**Parameters**

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

#### 5.44.3.7 void TypeExporter::SetGraph ( sparksee::gdb::Graph & *graph* ) [inherited]

Sets the graph that will be exported.

**Parameters**

| | |
|---|---|
| *graph* | [in] Graph. |

#### 5.44.3.8 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 |

Referenced by EdgeTypeExporter::EdgeTypeExporter(), and EdgeTypeExporter::SetHeadAttribute().

#### 5.44.3.9 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. |

#### 5.44.3.10 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 |

Referenced by EdgeTypeExporter::EdgeTypeExporter(), and EdgeTypeExporter::SetHeadPosition().

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

Sets the output data destination.

**Parameters**

| | |
|---|---|
| *rw* | [in] Input RowWriter. |

#### 5.44.3.12 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 |

Referenced by EdgeTypeExporter::EdgeTypeExporter(), and EdgeTypeExporter::SetTailAttribute().

#### 5.44.3.13 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 |

Referenced by EdgeTypeExporter::EdgeTypeExporter(), and EdgeTypeExporter::SetTailPosition().

#### 5.44.3.14 void TypeExporter::SetType ( sparksee::gdb::type_t *type* ) [inherited]

Sets the type to be exported.

**Parameters**

| | |
|---|---|
| *type* | [in] Type identifier. |

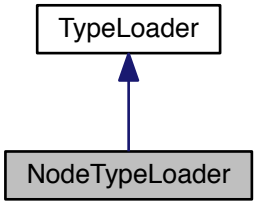
The documentation for this class was generated from the following file:

- NodeTypeExporter.h

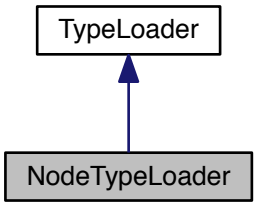
## 5.45 NodeTypeLoader Class Reference

NodeTypeLoader class.

Inheritance diagram for NodeTypeLoader:




Collaboration diagram for NodeTypeLoader:




### Public Member Functions

- NodeTypeLoader ()

  *Creates a new instance.*
- NodeTypeLoader (RowReader &rowReader, sparksee::gdb::Graph &graph, sparksee::gdb::type_t type, 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 &graph)

  *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 type)

  *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 &timestampFormat)

  *Sets a specific timestamp format.*
- void SetFrequency (sparksee::gdb::int32_t freq)

  *Sets the frequency of listener notification.*

#### Protected Types

#### Protected Member Functions

- void Run (Mode ph, sparksee::gdb::int32_t par) throw (sparksee::gdb::IOException, sparksee::gdb::Error)

  *Runs load process.*
- sparksee::gdb::bool_t CanRun ()

  *Checks that all the required settings are ready to run.*
- 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.*
- void SetTailMEP (sparksee::gdb::MissingEndpoint mep)

  *Sets the flag to create missing tail nodes when loading nodes or edges.*
- void SetHeadMEP (sparksee::gdb::MissingEndpoint mep)

  *Sets the flag to create missing head nodes when loading nodes or edges.*

#### 5.45.1 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

#### 5.45.2 Member Enumeration Documentation

##### 5.45.2.1 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.

#### 5.45.3 Constructor & Destructor Documentation

##### 5.45.3.1 NodeTypeLoader::NodeTypeLoader ( RowReader & *rowReader,* sparksee::gdb::Graph & *graph,* sparksee::gdb::type_t *type,* sparksee::gdb::AttributeList & *attrs,* sparksee::gdb::Int32List & *attrsPos* ) [inline]

Creates a new instance.

**Parameters**

| | |
|---|---|
| *rowReader* | [in] Input RowReader. |
| *graph* | [in] Graph. |
| *type* | [in] Type identifier. |
| *attrs* | [in] Attribute identifiers to be loaded. |
| *attrsPos* | [in] Attribute positions (column index >=0). |

#### 5.45.4 Member Function Documentation

#### 5.45.4.1 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.

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

Notifies progress to all registered listeners.

**Parameters**

| | |
|---|---|
| *ev* | [in] Progress event to be notified. |

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

Registers a new listener.

**Parameters**

| | | |
|---|---|---|
| in | *tel* | TypeLoaderListener to be registered. |

#### 5.45.4.4 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. |

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

Sets the list of attribute positions.

**Parameters**

| | |
|---|---|
| *attrsPos* | [in] Attribute positions (column index >=0). |

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

Sets the list of Attributes.

**Parameters**

| | |
|---|---|
| *attrs* | [in] Attribute identifiers to be loaded |

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

Sets the frequency of listener notification.

**Parameters**

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

#### 5.45.4.8 void TypeLoader::SetGraph ( sparksee::gdb::Graph & *graph* ) [inherited]

Sets the graph where the data will be loaded.

**Parameters**

| | |
|---|---|
| *graph* | [in] Graph. |

#### 5.45.4.9 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 |

Referenced by EdgeTypeLoader::EdgeTypeLoader(), and EdgeTypeLoader::SetHeadAttribute().

#### 5.45.4.10 void TypeLoader::SetHeadMEP ( sparksee::gdb::MissingEndpoint *mep* ) [protected], [inherited]

Sets the flag to create missing head nodes when loading nodes or edges.

**Parameters**

| | |
|---|---|
| *flag* | True if the nodes need to be created. False otherwise |

Referenced by EdgeTypeLoader::EdgeTypeLoader(), and EdgeTypeLoader::SetHeadMEP().

#### 5.45.4.11 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 |

Referenced by EdgeTypeLoader::EdgeTypeLoader(), and EdgeTypeLoader::SetHeadPosition().

#### 5.45.4.12 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. |

#### 5.45.4.13 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.

**Parameters**

| | |
|---|---|
| *path* | [in] The path to the error log file. |

**Exceptions**

| | |
|---|---|
| *IOException* | If bad things happen opening the file. |

#### 5.45.4.14 void TypeLoader::SetLogOff ( ) [inherited]

Truns 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.

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

Sets the input data source.

**Parameters**

| | |
|---|---|
| *rr* | [in] Input RowReader. |

#### 5.45.4.16 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 |

Referenced by EdgeTypeLoader::EdgeTypeLoader(), and EdgeTypeLoader::SetTailAttribute().

#### 5.45.4.17 void TypeLoader::SetTailMEP ( sparksee::gdb::MissingEndpoint *mep* ) [protected], [inherited]

Sets the flag to create missing tail nodes when loading nodes or edges.

**Parameters**

| | |
|---|---|
| *flag* | True if the nodes need to be created. False otherwise |

Referenced by EdgeTypeLoader::EdgeTypeLoader(), and EdgeTypeLoader::SetTailMEP().

#### 5.45.4.18 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 |

Referenced by EdgeTypeLoader::EdgeTypeLoader(), and EdgeTypeLoader::SetTailPosition().

#### 5.45.4.19 void TypeLoader::SetTimestampFormat ( const std::wstring & *timestampFormat* ) [inherited]

Sets a specific timestamp format.

**Parameters**

| | |
|---|---|
| *timestampFormat* | [in] A string with the timestamp format definition. |

#### 5.45.4.20 void TypeLoader::SetType ( sparksee::gdb::type_t *type* ) [inherited]

Sets the type to be loaded.

**Parameters**

| | |
|---|---|
| *type* | [in] Type identifier. |

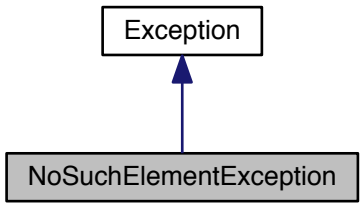
The documentation for this class was generated from the following file:

- NodeTypeLoader.h

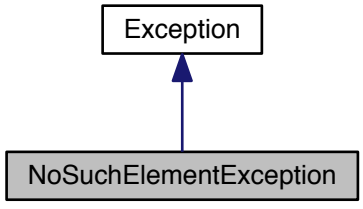
## 5.46 NoSuchElementException Class Reference

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.*

### 5.46.1 Detailed Description

No such element exception class.

#### Author

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

### 5.46.2 Constructor & Destructor Documentation

#### 5.46.2.1 NoSuchElementException::NoSuchElementException ( const std::string & *mess* )

Creates a new instance.

#### Parameters

| | |
|---|---|
| *mess* | [in] Message of the exception. |

### 5.46.3 Member Function Documentation

#### 5.46.3.1 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.

#### 5.46.3.2 void Exception::SetMessage ( const std::string & *mess* ) [inherited]

Sets the message of the exception.

**Parameters**

| | |
|---|---|
| *mess* | [in] Message. |

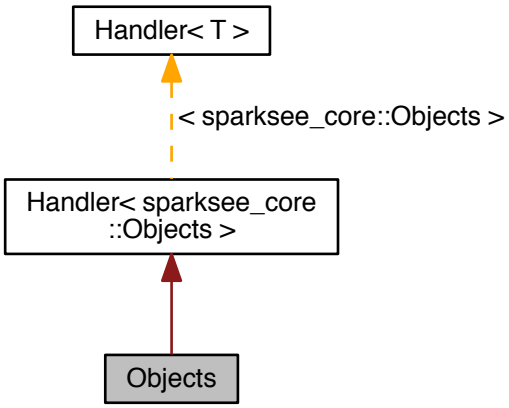
The documentation for this class was generated from the following file:

- Exception.h

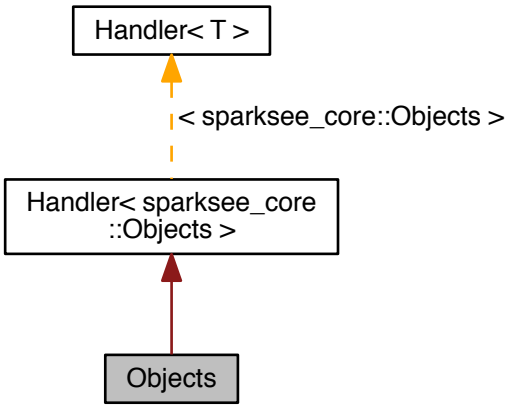
## 5.47 Objects Class Reference

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.*

#### Private Member Functions

- sparksee_core::Objects * GetHandler ()

  *Gets the handled reference.*
- const sparksee_core::Objects * GetHandler () const

  *Gets the handled reference.*
- void SetHandler (sparksee_core::Objects *h)

  *Sets the handled reference.*
- void FreeHandler ()

  *Frees (deletes) the handled reference.*
- bool_t IsNull () const

  *Gets if the handler is NULL.*

#### Friends

- class **Session**
- class **Graph**
- class **ObjectsIterator**

### 5.47.1 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

### 5.47.2 Member Function Documentation

#### 5.47.2.1 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.

#### 5.47.2.2 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. |

#### 5.47.2.3 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.

#### 5.47.2.4 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.

#### 5.47.2.5 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.

#### 5.47.2.6 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.

#### 5.47.2.7 Objects* Objects::Copy ( )

Creates a new Objects instance as a copy of the given one.

**Returns**

The new Objects instance.

#### 5.47.2.8 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.

#### 5.47.2.9 int64_t Objects::Count ( )

Gets the number of elements into the collection.

**Returns**

The number of elements into the collection.

#### 5.47.2.10 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.

#### 5.47.2.11 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.

#### 5.47.2.12 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.

#### 5.47.2.13 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.

#### 5.47.2.14 ObjectsIterator* Objects::Iterator ( )

Gets an ObjectsIterator.

**Returns**

ObjectsIterator instance.

#### 5.47.2.15 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.

#### 5.47.2.16 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.

#### 5.47.2.17 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.

#### 5.47.2.18 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.

#### 5.47.2.19 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.

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

- Objects.h

## 5.48 ObjectsIterator Class Reference

ObjectsIterator class.

### 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**

### 5.48.1 Detailed Description

ObjectsIterator class.

Iterator to traverse all the object identifiers from an Objects instance.

#### Author

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

### 5.48.2 Member Function Documentation

#### 5.48.2.1 bool_t ObjectsIterator::HasNext ( )

Gets if there are more elements to traverse.

#### Returns

TRUE if there are more elements to traverse, FALSE otherwise.

#### 5.48.2.2 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

## 5.49 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 numInvalidOIDs)
  *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.*

### 5.49.1 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

### 5.49.2 Constructor & Destructor Documentation

#### 5.49.2.1 OIDList::OIDList ( )

Constructor.

This creates an empty list.

#### 5.49.2.2 OIDList::OIDList ( int32_t *numInvalidOIDs* )

Constructor.

This creates a list with N invalid oids.

**Parameters**

| | |
|---|---|
| *numInvalidOIDs* | [in] The number of invalid oids added to the list. |

#### 5.49.2.3 OIDList::OIDList ( const std::vector< oid_t > & *v* )

Constructor.

**Parameters**

| | |
|---|---|
| *v* | [in] Vector. |

### 5.49.3 Member Function Documentation

#### 5.49.3.1 void OIDList::Add ( oid_t *attr* ) [inline]

Adds a Sparksee object identifier at the end of the list.

**Parameters**

| | |
|---|---|
| *attr* | [in] Sparksee object identifier. |

#### 5.49.3.2 int32_t OIDList::Count ( ) const [inline]

Number of elements in the list.

**Returns**

Number of elements in the list.

#### 5.49.3.3 OIDListIterator* OIDList::Iterator ( )

Gets a new OIDListIterator.

**Returns**

OIDListIterator instance.

#### 5.49.3.4 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

## 5.50 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**

### 5.50.1 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

### 5.50.2 Member Function Documentation

#### 5.50.2.1 bool_t OIDListIterator::HasNext ( ) [inline]

Gets if there are more elements.

#### Returns

TRUE if there are more elements, FALSE otherwise.

#### 5.50.2.2 oid_t OIDListIterator::Next ( ) [inline]

Moves to the next element.

**Returns**

The next element.

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

- Graph_data.h

## 5.51 PageRank Class Reference

PageRank class.

**Public Member Functions**

- PageRank (sparksee::gdb::Session &session)
  *Builds the PageRank.*
- virtual ~PageRank ()
  *Destructor.*
- void AddEdgeType (sparksee::gdb::type_t type, sparksee::gdb::EdgesDirection dir) throw (sparksee::gdb::Error)
  *Allows for traversing edges of the given type.*
- void AddAllEdgeTypes (sparksee::gdb::EdgesDirection dir)
  *Allows for traversing all edge types of the graph.*
- void AddNodeType (sparksee::gdb::type_t type) throw (sparksee::gdb::Error)
  *Allows for traversing nodes of the given type.*
- void AddAllNodeTypes ()
  *Allows for traversing all node types of the graph.*
- void SetNumIterations (sparksee::gdb::int32_t numIterations)
  *Sets the number of iterations to run the PageRank for.*
- void SetTolerance (sparksee::gdb::double64_t tolerance)
  *Sets the tolerance threashold to continue computing the PageRank after each iteration.*
- void SetDamping (sparksee::gdb::double64_t damping)
  *Sets the damping value for the PageRank.*
- void SetInitialPageRankValue (sparksee::gdb::double64_t startValue)
  *Sets the initial PageRank value.*
- void SetOutputAttributeType (sparksee::gdb::attr_t attr) throw (sparksee::gdb::Error)
  *Sets the output attribute type.*
- void SetEdgeWeightAttributeType (sparksee::gdb::attr_t attr) throw (sparksee::gdb::Error)
  *Sets the attribute to use as edge weight.*
- void SetDefaultWeight (sparksee::gdb::double64_t weight)
  *Sets the default weight for those cases when a given edge does not have a weight attribute set.*
- void SetStartingNode (sparksee::gdb::oid_t startNode) throw (sparksee::gdb::Error)
  *Sets the starting node of the page rank to compute the Personalized PageRank variant.*
- void Run () throw (sparksee::gdb::Error)
  *Runs the algorithm.*

### 5.51.1 Detailed Description

PageRank class.

Implements the PageRank algorithm

**Author**

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

### 5.51.2 Constructor & Destructor Documentation

#### 5.51.2.1 PageRank::PageRank ( sparksee::gdb::Session & *session* )

Builds the PageRank.

**Parameters**

| | |
|---|---|
| *session* | [in] The session to use |

### 5.51.3 Member Function Documentation

#### 5.51.3.1 void PageRank::AddAllEdgeTypes ( sparksee::gdb::EdgesDirection *dir* )

Allows for traversing all edge types of the graph.

The direction is interpreted as in which direction an edge can be followed from a node to influence other nodes.

**Parameters**

| | |
|---|---|
| *dir* | [in] Edge direction. |

#### 5.51.3.2 void PageRank::AddEdgeType ( sparksee::gdb::type_t *type*, sparksee::gdb::EdgesDirection *dir* ) throw sparksee::gdb::Error)

Allows for traversing edges of the given type.

If the edge type was already added, the existing direction is overwritten The direction is interpreted as in which direction an edge can be followed from a node to influence other nodes.

**Parameters**

| | |
|---|---|
| *type* | [in] Edge type. |
| *dir* | [in] Edge direction. |

#### 5.51.3.3 void PageRank::Run ( ) throw sparksee::gdb::Error)

Runs the algorithm.

**Exceptions**

| | |
|---|---|
| *sparksee::gdb::Error* | |

#### 5.51.3.4 void PageRank::SetDamping ( sparksee::gdb::double64_t *damping* )

Sets the damping value for the PageRank.

**Parameters**

| | |
|---|---|
| *damping* | [in] The damping value. Default: 0.85 |

#### 5.51.3.5 void PageRank::SetDefaultWeight ( sparksee::gdb::double64_t *weight* )

Sets the default weight for those cases when a given edge does not have a weight attribute set.

Default: 0.0

**Parameters**

| | |
|---|---|
| *weight* | [in] The default weight |

#### 5.51.3.6 void PageRank::SetEdgeWeightAttributeType ( sparksee::gdb::attr_t *attr* ) throw sparksee::gdb::Error)

Sets the attribute to use as edge weight.

If the multiple edge are set for traversal, this attribute must be of type GLOBAL_TYPE or EDGES_TYPE. Additionally, the attribute must be of type Double. Finally, negative weights are treated as non existing, so the default weight applies.

**Parameters**

| | |
|---|---|
| *attr* | [in] The attribute type to use as a weight. Default: InvalidAttribute |

**Exceptions**

| | |
|---|---|
| *sparksee::gdb::Error* | |

#### 5.51.3.7 void PageRank::SetInitialPageRankValue ( sparksee::gdb::double64_t *startValue* )

Sets the initial PageRank value.

If a starting node is set, this initial value is only set for the starting node and the rest of nodes are set to 0.0

**Parameters**

| | |
|---|---|
| *startValue* | [in] The initial value to set. Default: 0.0 |

#### 5.51.3.8 void PageRank::SetNumIterations ( sparksee::gdb::int32_t *numIterations* )

Sets the number of iterations to run the PageRank for.

**Parameters**

| | |
|---|---|
| *numIterations* | [in] The number of interations to set. Default: 20 |

#### 5.51.3.9 void PageRank::SetOutputAttributeType ( sparksee::gdb::attr_t *attr* ) throw sparksee::gdb::Error)

Sets the output attribute type.

If the PageRank will run on more than one node type, then the output attribute must be of type GLOBAL_TYPE or NODES_TYPE. Otherwise, it must be a valid attribute for the used node type.

**Parameters**

| | |
|---|---|
| *attr* | [in] The attribute to store the result. Default: InvalidAttribute |

#### 5.51.3.10 void PageRank::SetStartingNode ( sparksee::gdb::oid_t *startNode* ) throw sparksee::gdb::Error)

Sets the starting node of the page rank to compute the Personalized PageRank variant.

**Exceptions**

| | |
|---|---|
| *sparksee::gdb::Error* | |

#### 5.51.3.11 void PageRank::SetTolerance ( sparksee::gdb::double64_t *tolerance* )

Sets the tolerance threashold to continue computing the PageRank after each iteration.

If all the changes to any PPR value after an iteration are below that tolerance threshold, the algorithm finishes.

**Parameters**

| | |
|---|---|
| *tolerance* | [in] The tolerance to use normalized between 0 and 1. Default: 0.000001 |

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

- PageRank.h

## 5.52 Platform Class Reference

Platform class.

#### Static Public Member Functions

- static void GetStatistics (PlatformStatistics &stats)

  *Gets platform data and statistics.*

### 5.52.1 Detailed Description

Platform class.

#### Author

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

### 5.52.2 Member Function Documentation

#### 5.52.2.1 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

## 5.53 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 avialable (free) memory size in Bytes.*

#### Friends

- class **Platform**

### 5.53.1 Detailed Description

Platform data and statistics.

#### Author

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

### 5.53.2 Member Function Documentation

#### 5.53.2.1 int64_t PlatformStatistics::GetAvailableMem ( ) const [inline]

Gets avialable (free) memory size in Bytes.

#### Returns

Avialable (free) memory size in Bytes.

#### 5.53.2.2 int32_t PlatformStatistics::GetNumCPUs ( ) const [inline]

Gets the number of CPUs.

#### Returns

The number of CPUs.

#### 5.53.2.3 int64_t PlatformStatistics::GetRealTime ( ) const [inline]

Gets time in microseconds (since epoch).

#### Returns

Time in microseconds (since epoch).

#### 5.53.2.4 int64_t PlatformStatistics::GetSystemTime ( ) const [inline]

Gets CPU system time.

#### Returns

CPU system time.

#### 5.53.2.5 int64_t PlatformStatistics::GetTotalMem ( ) const [inline]

Gets physical memory size in Bytes.

**Returns**

Physical memory size in Bytes.

#### 5.53.2.6 int64_t PlatformStatistics::GetUserTime ( ) const [inline]

Gets CPU user time.

**Returns**

CPU user time.

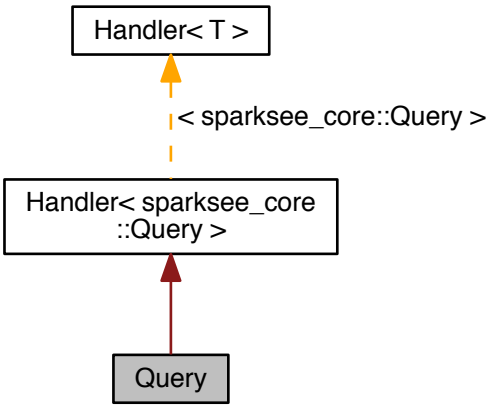
The documentation for this class was generated from the following file:

- common.h

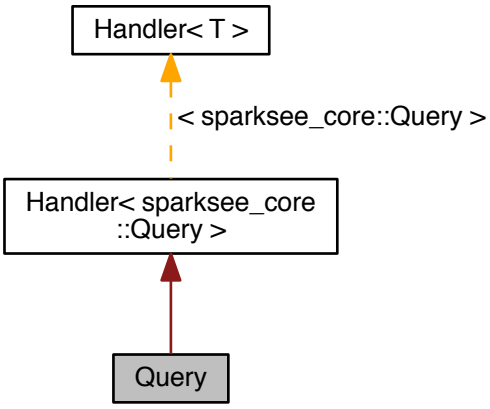
## 5.54 Query Class Reference

Query class.

Inheritance diagram for Query:



Collaboration diagram for Query:




#### Public Member Functions

- virtual ~Query ()

  *Destructor.*
- ResultSet * Execute (const std::wstring &stmt, bool_t reiterable=false)

  *Executes the given statement.*
- QueryStream * SetStream (const std::wstring &stream, QueryStream *handler)

  *Sets a query stream handler.*
- void SetDynamic (const std::wstring &name, Value &value)

  *Sets the value for a dynamic paramater.*

#### Private Member Functions

- sparksee_core::Query * GetHandler ()

  *Gets the handled reference.*
- const sparksee_core::Query * GetHandler () const

  *Gets the handled reference.*
- void SetHandler (sparksee_core::Query *h)

  *Sets the handled reference.*
- void FreeHandler ()

  *Frees (deletes) the handled reference.*
- bool_t IsNull () const

  *Gets if the handler is NULL.*

#### Friends

- class **Session**
- class **QueryContext**

#### 5.54.1 Detailed Description

Query class.

**Author**

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

#### 5.54.2 Member Function Documentation

##### 5.54.2.1 ResultSet* Query::Execute ( const std::wstring & *stmt*, bool_t *reiterable* = false )

Executes the given statement.

**Parameters**

| | |
|---|---|
| *stmt* | [in] Query statement. |
| *reiterable* | [in] Whether we want the resultset to be reiterable or not |

**Returns**

A ResultSet instance with the contents of the result of the query.

##### 5.54.2.2 void Query::SetDynamic ( const std::wstring & *name*, Value & *value* )

Sets the value for a dynamic paramater.

**Parameters**

| | |
|---|---|
| *name* | [in] Parameter name |
| *value* | [in] Parameter value |

##### 5.54.2.3 QueryStream* Query::SetStream ( const std::wstring & *stream*, QueryStream * *handler* )

Sets a query stream handler.

Query streams handlers are created and destroyed by the caller.

**Parameters**

| | |
|---|---|
| *stream* | [in] The stream name |
| *handler* | [in] Query stream handler |

**Returns**

The previous handler, or NULL if it does not exists

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

- Query.h

## 5.55 QueryContext Class Reference

Query context interface.

### Public Member Functions

- QueryContext ()

  *Default constructor.*
- virtual ~QueryContext ()

  *Destructor.*
- Query * NewQuery ()

  *Creates a new Query.*

### 5.55.1 Detailed Description

Query context interface.

A QueryContext contains and manages the resources required to run a Query. A Session is one example of a QueryContext connected to a Sparksee database. The applications can implement their own contexts to run queries out of Sparksee.

#### Author

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

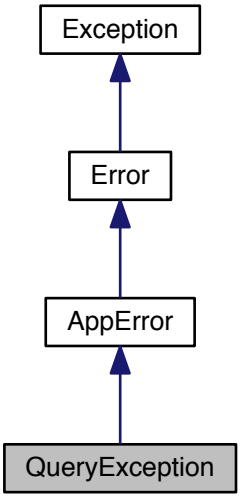
The documentation for this class was generated from the following file:

- QueryContext.h

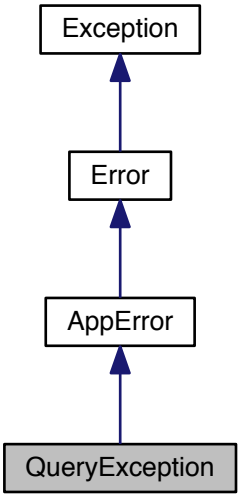
## 5.56 QueryException Class Reference

Query exception class.

Inheritance diagram for QueryException:




Collaboration diagram for QueryException:




**Public Member Functions**

- QueryException ()

  *Creates a new instance.*

- QueryException (const std::string &mess)

  *Creates a new instance.*
- QueryException (const sparksee_core::QueryError *core_error)

  *Creates a new instance.*
- QueryException (const sparksee_core::CypherError *cypher_error)

  *Creates a new instance.*
- virtual ~QueryException ()

  *Destructor.*
- sparksee::gdb::int32_t GetOffset ()

  *Get thhe Offset in the query that may have triggered the error.*
- 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 void ThrowError (int32_t coreErrorCode)

  *Throws a new Error instance from a sparksee_core error code.*

#### Protected Attributes

- sparksee::gdb::int32_t m_offset

  *The Offset in the query that triggered the error.*
- std::string message

  *Message of the exception.*

### 5.56.1 Detailed Description

Query exception class.

#### Author

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

### 5.56.2 Constructor & Destructor Documentation

#### 5.56.2.1 QueryException::QueryException ( const std::string & *mess* )

Creates a new instance.

#### Parameters

| | |
|---|---|
| *mess* | [in] Message of the exception. |

#### 5.56.2.2 QueryException::QueryException ( const sparksee_core::QueryError * *core_error* )

Creates a new instance.

**Parameters**

| | |
|---|---|
| *core_error* | [in] Dexcore SQLError. |

#### 5.56.2.3 QueryException::QueryException ( const sparksee_core::CypherError * *cypher_error* )

Creates a new instance.

**Parameters**

| | |
|---|---|
| *core_error* | [in] Dexcore SQLError. |

### 5.56.3 Member Function Documentation

#### 5.56.3.1 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.

#### 5.56.3.2 void Exception::SetMessage ( const std::string & *mess* ) [inherited]

Sets the message of the exception.

**Parameters**

| | |
|---|---|
| *mess* | [in] Message. |

#### 5.56.3.3 static void Error::ThrowError ( int32_t *coreErrorCode* ) [static], [inherited]

Throws 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 will throw an Error instance of an specific Error subclass instance.

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

- Exception.h

## 5.57 QueryStream Class Reference

Query stream interface.

#### Public Member Functions

- virtual ~QueryStream ()
  *Destructor.*
- virtual bool_t Prepare (const ValueList &list)=0
  *Prepares the stream before it is started.*
- virtual bool_t Start (ResultSetList &list)=0
  *Starts the stream.*
- virtual bool_t Fetch (ValueList &list)=0
  *Gets the next row and moves the iterator forward.*

#### Protected Member Functions

- QueryStream ()
  *Default constructor.*

### 5.57.1 Detailed Description

Query stream interface.

A QueryStream is the interface between the application and the STREAM operator. When the operator starts inside a Query, the method is prepared with query-defined arguments. Then, if there are input operations, the STREAM operator builds the ResultSets and starts the iteration. Finally, the operator fetches rows until no more are available.

Application exceptions must be cached by the subclass that implements the interface.

#### Author

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

### 5.57.2 Member Function Documentation

#### 5.57.2.1 virtual bool_t QueryStream::Fetch ( ValueList & *list* ) [pure virtual]

Gets the next row and moves the iterator forward.

The end of sequence is denoted by returning TRUE with an empty row. A valid row must contain as many values (even NULL) as expected by the query.

**Parameters**

| | |
|---|---|
| *list* | [out] Storage for the new rows |

**Returns**

TRUE if there is a row or end of sequence, FALSE on error

#### 5.57.2.2 virtual bool_t QueryStream::Prepare ( const ValueList & *list* ) [pure virtual]

Prepares the stream before it is started.

**Parameters**

| | |
|---|---|
| *list* | [in] Optional list of arguments |

**Returns**

FALSE on error

#### 5.57.2.3 virtual bool_t QueryStream::Start ( ResultSetList & *list* ) [pure virtual]

Starts the stream.

**Parameters**

| | |
|---|---|
| *list* | [in] Optional list of input ResultSets |

**Returns**

*FALSE* on error

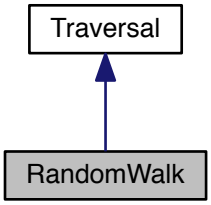
The documentation for this class was generated from the following file:

- Query.h

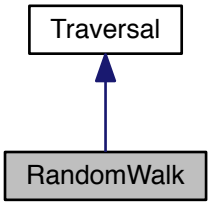
## 5.58 RandomWalk Class Reference

RandomWalk class.

Inheritance diagram for RandomWalk:




Collaboration diagram for RandomWalk:




#### Public Member Functions

- RandomWalk (sparksee::gdb::Session &session, sparksee::gdb::oid_t node)

  *Builds the RandomWalk.*
- virtual ~RandomWalk ()

  *Destructor.*
- virtual void AddEdgeType (sparksee::gdb::type_t type, sparksee::gdb::EdgesDirection dir)

  *Allows for traversing edges of the given type.*
- virtual void AddAllEdgeTypes (sparksee::gdb::EdgesDirection dir)

  *Allows for traversing all edge types of the graph.*
- virtual void AddNodeType (sparksee::gdb::type_t type)

  *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 ()

  *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.*
- void SetEdgeWeightAttributeType (sparksee::gdb::attr_t attr)

  *Sets the attribute to use as edge weight.*
- void SetDefaultWeight (sparksee::gdb::double64_t weight)

  *Sets the default weight for those cases when a given edge does not have a weight attribute set.*
- void Reset (sparksee::gdb::oid_t startNode)

  *Sets the starting node of the RandomWalk.*
- void SetReturnParameter (sparksee::gdb::double64_t val)

  *Sets the return parameter of the RandomWalk.*
- void SetInOutParameter (sparksee::gdb::double64_t val)

  *Sets the In-Out parameter of the RandomWalk.*
- void SetSeed (sparksee::gdb::int32_t seed)

  *Sets the seed of the random walk.*
- 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.*

### 5.58.1 Detailed Description

RandomWalk class.

Implements the RandomWalk algorithm

**Author**

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

### 5.58.2 Constructor & Destructor Documentation

#### 5.58.2.1 RandomWalk::RandomWalk ( sparksee::gdb::Session & *session*, sparksee::gdb::oid_t *node* )

Builds the RandomWalk.

**Parameters**

| | |
|---|---|
| *session* | [in] The session to use |
| *node* | [in] The starting node of the traversal |

### 5.58.3 Member Function Documentation

#### 5.58.3.1 virtual void RandomWalk::AddAllEdgeTypes ( sparksee::gdb::EdgesDirection *dir* ) [virtual]

Allows for traversing all edge types of the graph.

**Parameters**

| | |
|---|---|
| *dir* | [in] Edge direction. |

Reimplemented from Traversal.

#### 5.58.3.2 virtual void RandomWalk::AddEdgeType ( sparksee::gdb::type_t *type*, sparksee::gdb::EdgesDirection *dir* ) [virtual]

Allows for traversing edges of the given type.

If the edge type was already added, the existing direction is overwritten

**Parameters**

| | |
|---|---|
| *type* | [in] Edge type. |
| *dir* | [in] Edge direction. |

Reimplemented from Traversal.

#### 5.58.3.3 virtual void RandomWalk::AddNodeType ( sparksee::gdb::type_t *type* ) [virtual]

Allows for traversing nodes of the given type.

**Parameters**

| | |
|---|---|
| *type* | The node type to add |

Reimplemented from Traversal.

#### 5.58.3.4 virtual void RandomWalk::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. |

Reimplemented from Traversal.

#### 5.58.3.5 virtual void RandomWalk::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. |

Reimplemented from Traversal.

#### 5.58.3.6 virtual sparksee::gdb::int32_t RandomWalk::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.

#### 5.58.3.7 virtual sparksee::gdb::bool_t RandomWalk::HasNext ( ) [virtual]

Gets if there are more objects to be traversed.

**Returns**

TRUE if there are more objects, FALSE otherwise.

Implements Traversal.

#### 5.58.3.8 virtual sparksee::gdb::oid_t RandomWalk::Next ( ) [virtual]

Gets the next object of the traversal.

**Returns**

A node or edge identifier.

Implements Traversal.

#### 5.58.3.9 void RandomWalk::Reset ( sparksee::gdb::oid_t *startNode* )

Sets the starting node of the RandomWalk.

This method resets the RandomWalk.

**Exceptions**

| | |
|---|---|
| *sparksee::gdb::Error* | |

#### 5.58.3.10 void RandomWalk::SetDefaultWeight ( sparksee::gdb::double64_t *weight* )

Sets the default weight for those cases when a given edge does not have a weight attribute set.

Default: 0.0

**Parameters**

| | |
|---|---|
| *weight* | [in] The default weight |

#### 5.58.3.11 void RandomWalk::SetEdgeWeightAttributeType ( sparksee::gdb::attr_t *attr* )

Sets the attribute to use as edge weight.

If the multiple edge are set for traversal, this attribute must be of type GLOBAL_TYPE or EDGES_TYPE. Additionally, the attribute must be of type Double. Finally, negative weights are treated as non existing, so the default weight applies.

**Parameters**

| | |
|---|---|
| *attr* | [in] The attribute type to use as a weight. Default: InvalidAttribute |

#### 5.58.3.12 void RandomWalk::SetInOutParameter ( sparksee::gdb::double64_t *val* )

Sets the In-Out parameter of the RandomWalk.

**Parameters**

| | |
|---|---|
| *val* | The In-Out parameter to set. Default: 1.0 |

#### 5.58.3.13 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. |

#### 5.58.3.14 void RandomWalk::SetReturnParameter ( sparksee::gdb::double64_t *val* )

Sets the return parameter of the RandomWalk.

**Parameters**

| | |
|---|---|
| *val* | The return parameter to set. Default: 1.0 |

#### 5.58.3.15 void RandomWalk::SetSeed ( sparksee::gdb::int32_t *seed* )

Sets the seed of the random walk.

**Parameters**

| | |
|---|---|
| *seed* | The seed to generate the random numbers that drive the random walk |

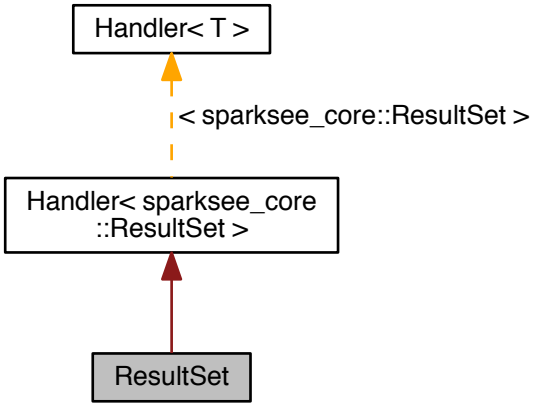
The documentation for this class was generated from the following file:

- RandomWalk.h

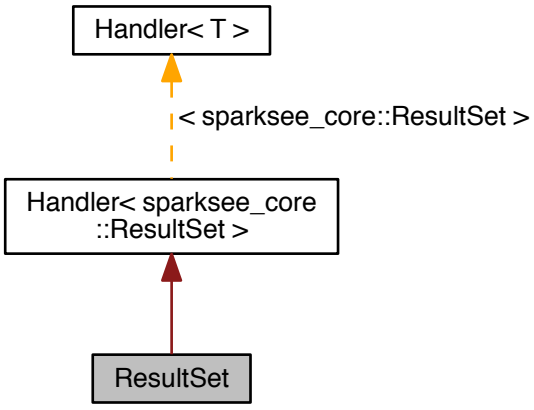
## 5.59 ResultSet Class Reference

ResultSet class.

Inheritance diagram for ResultSet:




Collaboration diagram for ResultSet:




### Public Member Functions

- virtual ~ResultSet ()

  *Destructor.*
- int32_t GetNumColumns () const

*Gets the number of columns.*

- const std::wstring & GetColumnName (int32_t index) const

  *Gets the name for the given column.*
- int32_t GetColumnIndex (const std::wstring &name) const

  *Gets the column index for the given column name.*
- DataType GetColumnDataType (int32_t index) const

  *Gets the datatype for the given column.*
- bool_t Next ()

  *Fetches the next row.*
- void GetColumn (int32_t index, Value &value) const

  *Gets the value for the given column.*
- Value * GetColumn (int32_t index) const

  *Gets the value for the given column.*
- void Rewind ()

  *Positions the cursor before the first row.*
- const std::wstring & GetJSON (int32_t rows)

  *Returns rows in JSON format.*

#### Private Member Functions

- sparksee_core::ResultSet * GetHandler ()

  *Gets the handled reference.*
- const sparksee_core::ResultSet * GetHandler () const

  *Gets the handled reference.*
- void SetHandler (sparksee_core::ResultSet *h)

  *Sets the handled reference.*
- void FreeHandler ()

  *Frees (deletes) the handled reference.*
- bool_t IsNull () const

  *Gets if the handler is NULL.*

#### Friends

- class **Query**

### 5.59.1 Detailed Description

ResultSet class.

#### Author

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

### 5.59.2 Member Function Documentation

#### 5.59.2.1 void ResultSet::GetColumn ( int32_t *index*, Value & *value* ) const

Gets the value for the given column.

**Parameters**

| | |
|---|---|
| *index* | [in] Column index. |
| *value* | [in\|out] Value. |

**Exceptions**

| | |
|---|---|
| *QueryException* | If a database access error occurs. |

#### 5.59.2.2 Value* ResultSet::GetColumn ( int32_t *index* ) const

Gets the value for the given column.

**Parameters**

| | |
|---|---|
| *index* | [in] Column index. |

**Returns**

The Value of the given column.

**Exceptions**

| | |
|---|---|
| *QueryException* | If a database access error occurs. |

#### 5.59.2.3 DataType ResultSet::GetColumnDataType ( int32_t *index* ) const

Gets the datatype for the given column.

**Parameters**

| | |
|---|---|
| *index* | [in] Column index. |

**Returns**

DataType for the given column.

#### 5.59.2.4 int32_t ResultSet::GetColumnIndex ( const std::wstring & *name* ) const

Gets the column index for the given column name.

**Parameters**

| | |
|---|---|
| *name* | [in] Column name. |

**Returns**

Column index.

#### 5.59.2.5 const std::wstring& ResultSet::GetColumnName ( int32_t *index* ) const

Gets the name for the given column.

**Parameters**

| | |
|---|---|
| *index* | [in] Column index. |

**Returns**

Column name.

#### 5.59.2.6 const std::wstring& ResultSet::GetJSON ( int32_t *rows* )

Returns rows in JSON format.

Rows are returned from the current position.

**Parameters**

| | |
|---|---|
| *rows* | [in] Maximum number of rows |

**Returns**

JSON representation of the next <rows> rows in the resultset

#### 5.59.2.7 int32_t ResultSet::GetNumColumns ( ) const

Gets the number of columns.

Columns are in the range [0...COLUMNS).

**Returns**

The number of columns.

#### 5.59.2.8 bool_t ResultSet::Next ( )

Fetches the next row.

A ResultSet cursor is initially positioned before the first row; the first call to the method "Next" makes the first row the current row; the second call makes the second row the current row, and so on.

**Returns**

TRUE if the next row has been successfully fetched, FALSE otherwise.

#### Exceptions

| | |
|---|---|
| *QueryException* | If a database access error occurs. |

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

- ResultSet.h

## 5.60 ResultSetList Class Reference

ResultSet list.

#### Public Member Functions

- int32_t Count () const
  *Number of elements in the list.*
- ResultSetList ()
  *Constructor.*
- ~ResultSetList ()
  *Destructor.*
- void Clear ()
  *Clears the list.*
- ResultSet * Get (int32_t index) const
  *Returns the ResultSet at the specified position in the list.*
- ResultSetListIterator * Iterator ()
  *Gets a new ResultSetListIterator.*

### 5.60.1 Detailed Description

ResultSet list.

It stores a ResultSet list.

#### Author

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

### 5.60.2 Constructor & Destructor Documentation

#### 5.60.2.1 ResultSetList::ResultSetList ( )

Constructor.

This creates an empty list.

### 5.60.3 Member Function Documentation

#### 5.60.3.1 int32_t ResultSetList::Count ( ) const

Number of elements in the list.

#### Returns

Number of elements in the list.

#### 5.60.3.2 ResultSet* ResultSetList::Get ( int32_t *index* ) const

Returns the ResultSet at the specified position in the list.

#### Parameters

| | |
|---|---|
| *index* | [in] Index of the element to return, starting at 0. |

#### 5.60.3.3 ResultSetListIterator* ResultSetList::Iterator ( )

Gets a new ResultSetListIterator.

#### Returns

ResultSetListIterator instance.

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

- ResultSet.h

## 5.61 ResultSetListIterator Class Reference

ResultSetList iterator class.

#### Public Member Functions

- ~ResultSetListIterator ()
  *Destructor.*
- const ResultSet * Next ()
  *Moves to the next element.*
- bool_t HasNext ()
  *Gets if there are more elements.*

#### Friends

- class **ResultSetList**

### 5.61.1 Detailed Description

ResultSetList iterator class.

Iterator to traverse all the values into a ResultSetList instance.

#### Author

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

#### 5.61.2 Member Function Documentation

##### 5.61.2.1 bool_t ResultSetListIterator::HasNext ( )

Gets if there are more elements.

**Returns**

TRUE if there are more elements, FALSE otherwise.

##### 5.61.2.2 const ResultSet* ResultSetListIterator::Next ( )

Moves to the next element.

**Returns**

The next element.

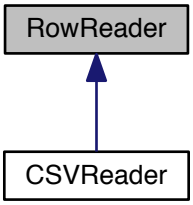
The documentation for this class was generated from the following file:

- ResultSet.h

## 5.62 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.*

### 5.62.1 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

### 5.62.2 Constructor & Destructor Documentation

#### 5.62.2.1 RowReader::RowReader ( ) [inline],[protected]

Empty constructor.

Protected because no one should instantiate a RowReader. Just inherited classes can use this empty constructor.

References END_SPARKSEE_GDB_NAMESPACE.

### 5.62.3 Member Function Documentation

#### 5.62.3.1 virtual void RowReader::Close ( ) throw sparksee::gdb::IOException) [pure virtual]

Closes the reader.

#### Exceptions

| | |
|---|---|
| *IOException* | If the close fails. |

Implemented in CSVReader.

#### 5.62.3.2 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.

#### 5.62.3.3 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.

#### 5.62.3.4 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.

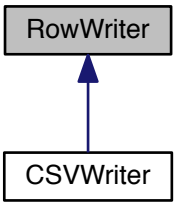
The documentation for this class was generated from the following file:

- RowReader.h

## 5.63 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.*

### 5.63.1 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

### 5.63.2 Constructor & Destructor Documentation

#### 5.63.2.1 RowWriter::RowWriter ( ) [inline], [protected]

Empty constructor.

Protected because no one should instantiate a RowWriter. Just inherited classes can use this empty constructor.

References END_SPARKSEE_IO_NAMESPACE.

#### 5.63.3 Member Function Documentation

##### 5.63.3.1 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.

##### 5.63.3.2 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.

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

- RowWriter.h

## 5.64 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.*

### 5.64.1 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

### 5.64.2 Member Function Documentation

#### 5.64.2.1 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 writen. |
| *db* | [in] Database. |

#### Exceptions

| | |
|---|---|
| *IOException* | If bad things happen opening or writing the file. |

#### 5.64.2.2 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.

#### 5.64.2.3 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. |

#### 5.64.2.4 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. |

#### 5.64.2.5 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. |

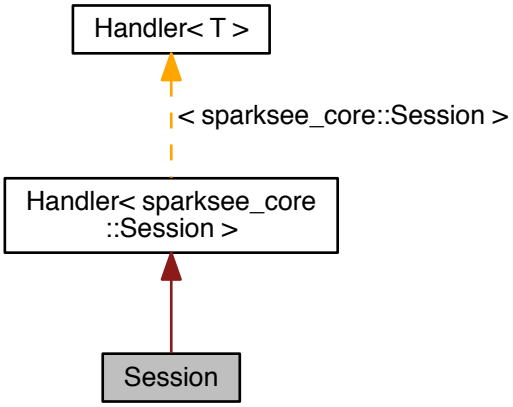
The documentation for this class was generated from the following file:

- ScriptParser.h

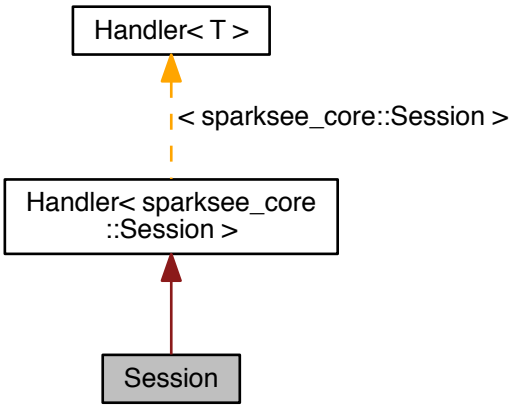
## 5.65 Session Class Reference

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 BeginUpdate ()

  *Begins an update transaction.*
- void Commit ()

  *Commits a transaction.*
- void Rollback ()

  *Rollbacks a transaction.*
- Query * NewQuery (sparksee::gdb::QueryLanguage lang)

  *Creates a new Query.*
- int64_t PreCommit ()

  *PreCommits a transaction.*
- int64_t GetInMemoryPoolCapacity ()

  *Gets the capacity of the in-memory pool.*

### Private Member Functions

- sparksee_core::Session * GetHandler ()

  *Gets the handled reference.*
- const sparksee_core::Session * GetHandler () const

  *Gets the handled reference.*
- void SetHandler (sparksee_core::Session *h)

  *Sets the handled reference.*
- void FreeHandler ()

  *Frees (deletes) the handled reference.*
- bool_t IsNull () const

  *Gets if the handler is NULL.*

### Friends

- class **Database**
- class **Graph**
- class **Objects**
- class **ObjectsIterator**
- class **KeyValues**
- class **Values**
- class **ValuesIterator**
- class **TextStream**
- class **ValueArray**

#### 5.65.1 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

#### 5.65.2 Member Function Documentation

##### 5.65.2.1 Graph* Session::GetGraph ( )

Gets the Graph instance.

**Returns**

The Graph instance.

##### 5.65.2.2 int64_t Session::GetInMemoryPoolCapacity ( )

Gets the capacity of the in-memory pool.

**Returns**

Returns the capacity of the in-memory pool

##### 5.65.2.3 Objects* Session::NewObjects ( )

Creates a new Objects instance.

**Returns**

The new Objects instance.

##### 5.65.2.4 Query* Session::NewQuery ( sparksee::gdb::QueryLanguage *lang* )

Creates a new Query.

**Parameters**

| | |
|---|---|
| *lang* | The query language to create the query for |

#### 5.65.2.5 int64_t Session::PreCommit ( )

PreCommits a transaction.

YOU SHOULD NOT USE THIS METHOD. This method exists for a specific service and it is used to force that the transaction is written in the recovery log even though it had not been committed yet (and may never be).

**Returns**

The transaction id. This has to be used in RedoPrecommited in case of a recovery process.

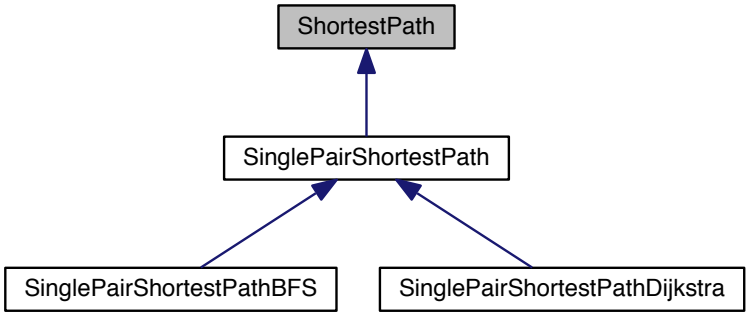
The documentation for this class was generated from the following file:

- Session.h

## 5.66 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 type, sparksee::gdb::EdgesDirection dir)

  *Allows for traversing edges of the given type.*
- virtual void AddAllEdgeTypes (sparksee::gdb::EdgesDirection dir)

  *Allows for traversing all edge types of the graph.*
- virtual void AddNodeType (sparksee::gdb::type_t type)

  *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.*

### 5.66.1 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

### 5.66.2 Constructor & Destructor Documentation

#### 5.66.2.1 ShortestPath::ShortestPath ( sparksee::gdb::Session & *s* ) [protected]

Creates a new instance.

#### Parameters

| | |
|---|---|
| *s* | [in] Session to get the graph from and perform traversal. |

### 5.66.3 Member Function Documentation

#### 5.66.3.1 virtual void ShortestPath::AddAllEdgeTypes ( sparksee::gdb::EdgesDirection *dir* ) [virtual]

Allows for traversing all edge types of the graph.

**Parameters**

| | |
|---|---|
| *dir* | [in] Edge direction. |

#### 5.66.3.2 virtual void ShortestPath::AddEdgeType ( sparksee::gdb::type_t *type*, sparksee::gdb::EdgesDirection *dir* ) [virtual]

Allows for traversing edges of the given type.

**Parameters**

| | |
|---|---|
| *type* | [in] Edge type. |
| *dir* | [in] Edge direction. |

#### 5.66.3.3 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. |

#### 5.66.3.4 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. |

#### 5.66.3.5 virtual void ShortestPath::Run ( ) [pure virtual]

Runs the algorithm.

This method can only be called once.

Implemented in SinglePairShortestPathDijkstra, and SinglePairShortestPathBFS.

#### 5.66.3.6 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**

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

### 5.66.4 Member Data Documentation

#### 5.66.4.1 sparksee::gdb::int32_t ShortestPath::maxHops [protected]

Maximum hops restriction.

It must be positive or zero. Zero means unlimited.

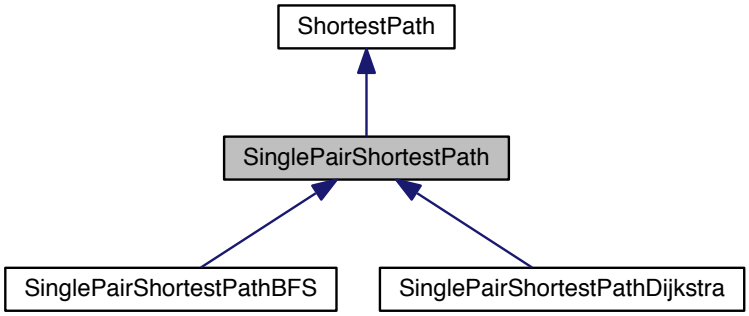
The documentation for this class was generated from the following file:

- ShortestPath.h

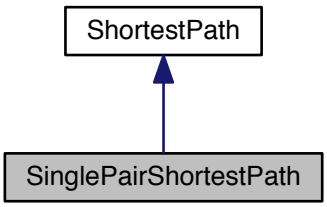
## 5.67 SinglePairShortestPath Class Reference

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 type, sparksee::gdb::EdgesDirection dir)

  *Allows for traversing edges of the given type.*
- virtual void AddAllEdgeTypes (sparksee::gdb::EdgesDirection dir)

  *Allows for traversing all edge types of the graph.*
- virtual void AddNodeType (sparksee::gdb::type_t type)

  *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.*

#### 5.67.1 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

#### 5.67.2 Constructor & Destructor Documentation

##### 5.67.2.1 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. |

#### 5.67.3 Member Function Documentation

##### 5.67.3.1 virtual void ShortestPath::AddAllEdgeTypes ( sparksee::gdb::EdgesDirection *dir* ) [virtual], [inherited]

Allows for traversing all edge types of the graph.

**Parameters**

| | |
|---|---|
| *dir* | [in] Edge direction. |

##### 5.67.3.2 virtual void ShortestPath::AddEdgeType ( sparksee::gdb::type_t *type*, sparksee::gdb::EdgesDirection *dir* ) [virtual], [inherited]

Allows for traversing edges of the given type.

**Parameters**

| | |
|---|---|
| *type* | [in] Edge type. |
| *dir* | [in] Edge direction. |

#### 5.67.3.3 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. |

#### 5.67.3.4 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. |

#### 5.67.3.5 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 algoritms, the cost is the sum of the costs of the edges of the shortest path.

**Returns**

The cost of the shortest path.

Implemented in SinglePairShortestPathDijkstra, and SinglePairShortestPathBFS.

#### 5.67.3.6 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 SinglePairShortestPathDijkstra, and SinglePairShortestPathBFS.

#### 5.67.3.7 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 SinglePairShortestPathDijkstra, and SinglePairShortestPathBFS.

#### 5.67.3.8 virtual void ShortestPath::Run ( ) [pure virtual], [inherited]

Runs the algorithm.

This method can only be called once.

Implemented in SinglePairShortestPathDijkstra, and SinglePairShortestPathBFS.

#### 5.67.3.9 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**

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

### 5.67.4 Member Data Documentation

#### 5.67.4.1 sparksee::gdb::int32_t ShortestPath::maxHops [protected], [inherited]

Maximum hops restriction.

It must be positive or zero. Zero means unlimited.

#### 5.67.4.2 sparksee::gdb::OIDList* SinglePairShortestPath::pathAsEdges [protected]

Ordered set of edge identifiers representing the shortest path.

May be computed lazily when requested from the pathAsNodes.

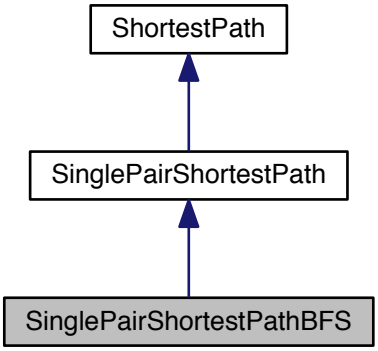
The documentation for this class was generated from the following file:

- SinglePairShortestPath.h

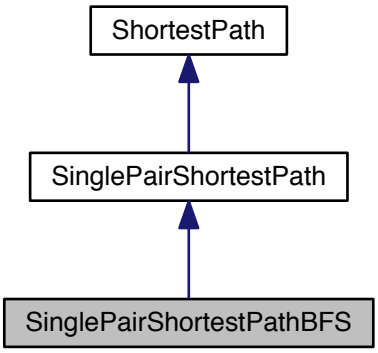
## 5.68 SinglePairShortestPathBFS Class Reference

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 &session, 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 type, sparksee::gdb::EdgesDirection dir)

  *Allows for traversing edges of the given type.*
- virtual void AddAllEdgeTypes (sparksee::gdb::EdgesDirection dir)

  *Allows for traversing all edge types of the graph.*
- virtual void AddNodeType (sparksee::gdb::type_t type)

  *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.*

### 5.68.1 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

### 5.68.2 Constructor & Destructor Documentation

#### 5.68.2.1 SinglePairShortestPathBFS::SinglePairShortestPathBFS ( sparksee::gdb::Session & *session*, sparksee::gdb::oid_t *src*, sparksee::gdb::oid_t *dst* )

Creates a new instance.

**Parameters**

| | |
|---|---|
| *session* | [in] Session to get the graph from and perform traversal. |
| *src* | [in] Source node. |
| *dst* | [dst] Destination node. |

### 5.68.3 Member Function Documentation

#### 5.68.3.1 virtual void ShortestPath::AddAllEdgeTypes ( sparksee::gdb::EdgesDirection *dir* ) [virtual], [inherited]

Allows for traversing all edge types of the graph.

**Parameters**

| | |
|---|---|
| *dir* | [in] Edge direction. |

#### 5.68.3.2 virtual void ShortestPath::AddEdgeType ( sparksee::gdb::type_t *type,* sparksee::gdb::EdgesDirection *dir* ) [virtual], [inherited]

Allows for traversing edges of the given type.

**Parameters**

| | |
|---|---|
| *type* | [in] Edge type. |
| *dir* | [in] Edge direction. |

#### 5.68.3.3 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.

#### 5.68.3.4 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. |

#### 5.68.3.5 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. |

#### 5.68.3.6 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.

#### 5.68.3.7 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.

#### 5.68.3.8 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.

#### 5.68.3.9 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**

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

#### 5.68.4 Member Data Documentation

##### 5.68.4.1 sparksee::gdb::int32_t ShortestPath::maxHops [protected], [inherited]

Maximum hops restriction.

It must be positive or zero. Zero means unlimited.

##### 5.68.4.2 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.

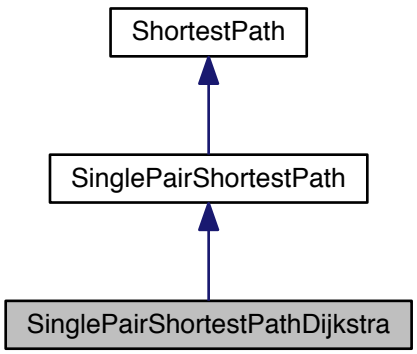
The documentation for this class was generated from the following file:

- SinglePairShortestPathBFS.h

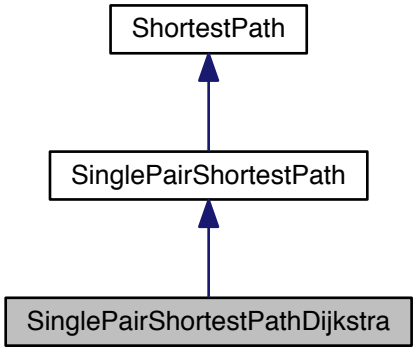
### 5.69 SinglePairShortestPathDijkstra Class Reference

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 &session, sparksee::gdb::oid_t src, sparksee::gdb::oid_t dst)

  *Creates a new instance.*
- virtual void AddWeightedEdgeType (sparksee::gdb::type_t type, sparksee::gdb::EdgesDirection dir, 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 void SetDynamicEdgeCostCallback (SinglePairShortestPathDijkstraDynamicCostCalculator *dynCost)

  *Set a class callback to dynamically calculate the cost of the 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 type, sparksee::gdb::EdgesDirection dir)

  *Allows for traversing edges of the given type.*
- virtual void AddAllEdgeTypes (sparksee::gdb::EdgesDirection dir)

  *Allows for traversing all edge types of the graph.*
- virtual void AddNodeType (sparksee::gdb::type_t type)

*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.*

#### 5.69.1 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

#### 5.69.2 Constructor & Destructor Documentation

##### 5.69.2.1 SinglePairShortestPathDijkstra::SinglePairShortestPathDijkstra ( sparksee::gdb::Session & *session*, sparksee::gdb::oid_t *src*, sparksee::gdb::oid_t *dst* )

Creates a new instance.

**Parameters**

| | |
|---|---|
| *session* | [in] Session to get the graph from and perform traversal. |
| *src* | [in] Source node. |
| *dst* | [dst] Destination node. |

#### 5.69.3 Member Function Documentation

##### 5.69.3.1 virtual void ShortestPath::AddAllEdgeTypes ( sparksee::gdb::EdgesDirection *dir* ) [virtual], [inherited]

Allows for traversing all edge types of the graph.

**Parameters**

| | |
|---|---|
| *dir* | [in] Edge direction. |

#### 5.69.3.2 virtual void ShortestPath::AddEdgeType ( sparksee::gdb::type_t *type*, sparksee::gdb::EdgesDirection *dir* ) [virtual], [inherited]

Allows for traversing edges of the given type.

**Parameters**

| | |
|---|---|
| *type* | [in] Edge type. |
| *dir* | [in] Edge direction. |

#### 5.69.3.3 virtual void SinglePairShortestPathDijkstra::AddWeightedEdgeType ( sparksee::gdb::type_t *type*, sparksee::gdb::EdgesDirection *dir*, sparksee::gdb::attr_t *attr* ) [virtual]

Allows for traversing edges of the given type using the given attribute as the weight.

**Parameters**

| | |
|---|---|
| *type* | [in] Edge type. |
| *dir* | [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. |

#### 5.69.3.4 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. |

#### 5.69.3.5 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. |

#### 5.69.3.6 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.

#### 5.69.3.7 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.

#### 5.69.3.8 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.

#### 5.69.3.9 virtual void SinglePairShortestPathDijkstra::SetDynamicEdgeCostCallback ( SinglePairShortestPathDijkstra↩ DynamicCost * *dynCostCalculator* ) [virtual]

Set a class callback to dynamically calculate the cost of the edges.

The callback can be set to NULL (the default) to use the normal attribute based cost weights. The given class must be kept alive by the user for as long as the algorithm is running.

**Parameters**

| | |
|---|---|
| *dynCostCalculator* | [in] Class callback to calculate the edge costs |

#### 5.69.3.10 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**

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

#### 5.69.3.11 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.

**Parameters**

| | |
|---|---|
| *weight* | [in] The weight value for unweighted edges. |

### 5.69.4 Member Data Documentation

#### 5.69.4.1 sparksee::gdb::int32_t ShortestPath::maxHops [protected], [inherited]

Maximum hops restriction.

It must be positive or zero. Zero means unlimited.

#### 5.69.4.2 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.

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

- SinglePairShortestPathDijkstra.h

## 5.70 SinglePairShortestPathDijkstraDynamicCost Class Reference

Defines how to calculate an edge weight.

**Public Member Functions**

- virtual ~SinglePairShortestPathDijkstraDynamicCost ()

  *Destructor.*
- virtual sparksee::gdb::double64_t CalculateEdgeCost (sparksee::gdb::oid_t sourceNode, sparksee::gdb::double64_t sourceCost, sparksee::gdb::int32_t sourceLevel, sparksee::gdb::oid_t targetNode, sparksee::gdb::oid_t edge, sparksee::gdb::attr_t edgeWeightAttr)=0

  *Dynamic calculation of an edge weight.*

### 5.70.1 Detailed Description

Defines how to calculate an edge weight.

This is an interface which must be implemented by the user. While the algorithm is running, one or more calls for each edge that can be in the shortest path will be issued to get the weight of the edge that could change based on the predecessor node and the current minimum path from the source.

**Author**

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

### 5.70.2 Member Function Documentation

#### 5.70.2.1 virtual sparksee::gdb::double64_t SinglePairShortestPathDijkstraDynamicCost::CalculateEdgeCost ( sparksee::gdb::oid_t *sourceNode*, sparksee::gdb::double64_t *sourceCost*, sparksee::gdb::int32_t *sourceLevel*, sparksee::gdb::oid_t *targetNode*, sparksee::gdb::oid_t *edge*, sparksee::gdb::attr_t *edgeWeightAttr* ) [pure virtual]

Dynamic calculation of an edge weight.

This method will be called several times during the algorithm execution. It can be called for the same edge multiple times because the path to the source node (and as a consequence the source weight and levels) might change while the algorithm is running. This method will have the normal default implementation that only uses the specified edge attribute to get the weight or the unweighted edge cost if the edge doesn't have the specified cost attribute. But it can be overriden in a derived user class to implement a dinamic edge weight, that may need the sourceCost and sourceLevels to calculate the weight.

**Parameters**

| | |
|---|---|
| *sourceNode* | [in] The current node |
| *sourceCost* | [in] Current total cost up to the source node |
| *sourceLevel* | [in] Current path size up to the source node |
| *targetNode* | [in] The next node |
| *edge* | [in] The edge to the next node that this methos must weight |
| *edgeWeightAttr* | [in] The attribute that stores the edge weight or InvalidAttribute |

**Returns**

Returns the current weight of the edge (>=0) or <0 if the edge can not be used.

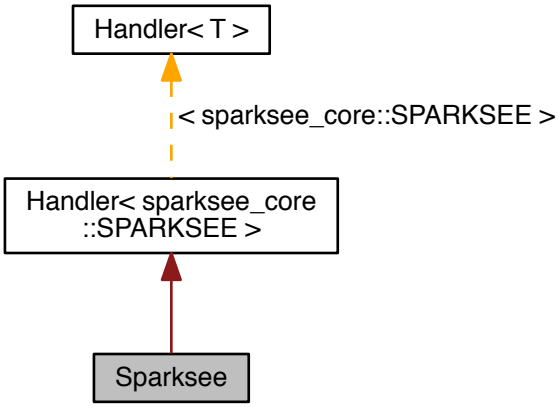
The documentation for this class was generated from the following file:

- SinglePairShortestPathDijkstra.h

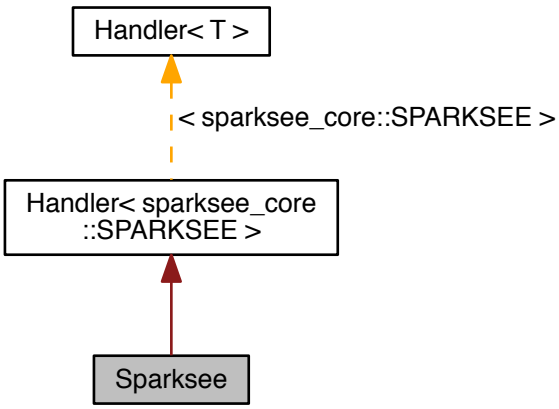
## 5.71 Sparksee Class Reference

Sparksee class.

Inheritance diagram for Sparksee:




Collaboration diagram for Sparksee:




**Public Member Functions**

- Sparksee (SparkseeConfig &config)

  *Creates a new instance.*
- virtual ~Sparksee ()

  *Destructor.*
- Database * Create (const std::wstring &path, const std::wstring &alias) throw (sparksee::gdb::FileNotFound↩ Exception, sparksee::gdb::Error)

  *Creates a new Database instance.*

- Database * Create (const std::wstring &path, const std::wstring &alias, const SparkseeConfig &config) throw (sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)

  *Creates a new Database instance.*
- Database * Open (const std::wstring &path, bool_t readOnly) throw (sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)

  *Opens an existing Database instance.*
- Database * Open (const std::wstring &path, bool_t readOnly, const SparkseeConfig &config) 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::File↩ NotFoundException, sparksee::gdb::Error)

  *Restores a Database from a backup file.*
- Database * Restore (const std::wstring &path, const std::wstring &backupFile, const SparkseeConfig &config) throw (sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)

  *Restores a Database from a backup file.*
- Database * RestoreEncryptedBackup (const std::wstring &path, const std::wstring &backupFile, const std↩ ::wstring &keyInHex, const std::wstring &ivInHex) throw (sparksee::gdb::FileNotFoundException, sparksee↩ ::gdb::Error)

  *Restores a Database from an encrypted backup file.*
- Database * RestoreEncryptedBackup (const std::wstring &path, const std::wstring &backupFile, const SparkseeConfig &config, const std::wstring &keyInHex, const std::wstring &ivInHex) throw (sparksee::gdb↩ ::FileNotFoundException, sparksee::gdb::Error)

  *Restores a Database from an encrypted backup file.*
- bool_t VerifyChecksums (const std::wstring &path) throw (sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)

  *Verifies the Checksum integrity of the given image file and it's recovery log file.*
- bool_t AddChecksums (const std::wstring &path) throw (sparksee::gdb::FileNotFoundException, sparksee↩ ::gdb::Error)

  *Converts a database WITHOUT checksums into a database WITH checksums.*
- bool_t RemoveChecksums (const std::wstring &path) throw (sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)

  *Converts a database WITH checksums into a database WITHOUT checksums.*
- void SetUnrecoverableErrorCallback (SPARKSEE_UNRECOVERABLE_ERR_CALLBACK callback)

  *Sets the unrecoverable error callback.*
- bool_t Encrypt (const std::wstring &path) throw (sparksee::gdb::FileNotFoundException, sparksee::gdb::↩ Error)

  *Converts a database WITHOUT encryption into an encrypted database.*
- bool_t Decrypt (const std::wstring &path) throw (sparksee::gdb::FileNotFoundException, sparksee::gdb::↩ Error)

  *Converts a database WITH encryption into a database WITHOUT encryption.*
- void ResizeInMemoryPool (int64_t newCapacity) throw (sparksee::gdb::SystemError)

  *Resizes the in memory pool allocator.*
- int64_t GetInMemoryPoolCapacity ()

  *Gets the capacity of the in-memory pool (in Megabytes)*

#### Static Public Attributes

- static const std::wstring Version

  *Sparksee version.*

#### Private Member Functions

- sparksee_core::SPARKSEE * GetHandler ()

  *Gets the handled reference.*
- const sparksee_core::SPARKSEE * GetHandler () const

  *Gets the handled reference.*
- void SetHandler (sparksee_core::SPARKSEE *h)

  *Sets the handled reference.*
- void FreeHandler ()

  *Frees (deletes) the handled reference.*
- bool_t IsNull () const

  *Gets if the handler is NULL.*

### 5.71.1 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

### 5.71.2 Constructor & Destructor Documentation

#### 5.71.2.1 Sparksee::Sparksee ( SparkseeConfig & *config* )

Creates a new instance.

#### Parameters

| | |
|---|---|
| *config* | [in/out] Sparksee configuration (may be updated). |

### 5.71.3 Member Function Documentation

#### 5.71.3.1 bool_t Sparksee::AddChecksums ( const std::wstring & *path* ) throw sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)

Converts a database WITHOUT checksums into a database WITH checksums.

Any error found is logged.

#### Precondition

The database (and it's recovery log if present) does NOT have checksums.

**Parameters**

| | |
|---|---|
| *path* | The path to the database image file |

**Returns**

Returns true if the checksum could be added or false otherwise.

**Exceptions**

| | |
|---|---|
| *sparksee::gdb::FileNotFoundException* | |
| *sparksee::gdb::Error* | |

#### 5.71.3.2 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. |

#### 5.71.3.3 Database* Sparksee::Create ( const std::wstring & *path*, const std::wstring & *alias*, const SparkseeConfig & *config* ) throw sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)

Creates a new Database instance.

**Parameters**

| | |
|---|---|
| *path* | [in] Database storage file. |
| *alias* | [in] Database alias name. |
| *config* | [in] Use a specific configuration instead of the one in this Sparksee |

**Returns**

A Database instance.

**Exceptions**

| | |
|---|---|
| *FileNotFoundException* | If the given file cannot be created. |

#### 5.71.3.4 bool_t Sparksee::Decrypt ( const std::wstring & *path* ) throw sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)

Converts a database WITH encryption into a database WITHOUT encryption.

Any error found is logged.

**Precondition**

The database (and it's recovery log if present) is encrypted.
The current encryption had been configured using SparkseeConfig.

**Parameters**

| | |
|---|---|
| *path* | The path to the database image file |

**Returns**

Returns true if the database could be unencrypted. The errors are returned with an exception.

**Exceptions**

| | |
|---|---|
| *sparksee::gdb::FileNotFoundException* | |
| *sparksee::gdb::Error* | |

#### 5.71.3.5 bool_t Sparksee::Encrypt ( const std::wstring & *path* ) throw sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)

Converts a database WITHOUT encryption into an encrypted database.

Any error found is logged.

**Precondition**

The database (and it's recovery log if present) does is NOT encrypted.
The encryption had already been configured using SparkseeConfig.

**Parameters**

| | |
|---|---|
| *path* | The path to the database image file |

**Returns**

Returns true if the database could be encrypted. The errors are returned with an exception.

**Exceptions**

| | |
|---|---|
| *sparksee::gdb::FileNotFoundException* | |
| *sparksee::gdb::Error* | |

#### 5.71.3.6 int64_t Sparksee::GetInMemoryPoolCapacity ( )

Gets the capacity of the in-memory pool (in Megabytes)

**Returns**

Returns the capacity of the in memory pool

#### 5.71.3.7 Database* Sparksee::Open ( const std::wstring & *path*, bool_t *readOnly* ) throw sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)

Opens an existing Database instance.

**Parameters**

| | |
|---|---|
| *path* | [in] Database storage file. |
| *readOnly* | [in] If TRUE, open Database in read-only mode. |

**Returns**

A Database instance.

**Exceptions**

| | |
|---|---|
| *FileNotFoundException* | If the given file does not exist. |

#### 5.71.3.8 Database* Sparksee::Open ( const std::wstring & *path*, bool_t *readOnly*, const SparkseeConfig & *config* ) throw sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)

Opens an existing Database instance.

The provided configuration will be used for all the database settings but not for the license, which must already be properly setup in this class.

**Parameters**

| | |
|---|---|
| *path* | [in] Database storage file. |
| *readOnly* | [in] If TRUE, open Database in read-only mode. |
| *config* | [in] Use a specific configuration instead of the one in this Sparksee |

**Returns**

A Database instance.

**Exceptions**

| | |
|---|---|
| *FileNotFoundException* | If the given file does not exist. |

#### 5.71.3.9 bool_t Sparksee::RemoveChecksums ( const std::wstring & *path* ) throw sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)

Converts a database WITH checksums into a database WITHOUT checksums.

Any error found is logged.

**Precondition**

The database (and it's recovery log if present) have checksums.

**Parameters**

| | |
|---|---|
| *path* | The path to the database image file |

**Returns**

Returns true if the checksum could be removed. The errors are returned with an exception.

**Exceptions**

| | |
|---|---|
| *sparksee::gdb::FileNotFoundException* | |
| *sparksee::gdb::Error* | |

#### 5.71.3.10 void Sparksee::ResizeInMemoryPool ( int64_t *newCapacity* ) throw sparksee::gdb::SystemError)

Resizes the in memory pool allocator.

**Parameters**

| | |
|---|---|
| *newCapacity* | The new capacity of the in memory pool allocator |

**Returns**

Returns true if was successful

#### 5.71.3.11 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. |

#### 5.71.3.12 Database* Sparksee::Restore ( const std::wstring & *path*, const std::wstring & *backupFile*, const SparkseeConfig & *config* ) 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. |
| *config* | [in] Use a specific configuration instead of the one in this Sparksee |

**Returns**

A Database instance.

**Exceptions**

| | |
|---|---|
| *FileNotFoundException* | If the given file cannot be created, or the exported data file does not exists. |

#### 5.71.3.13 Database* Sparksee::RestoreEncryptedBackup ( const std::wstring & *path*, const std::wstring & *backupFile*, const std::wstring & *keyInHex*, const std::wstring & *ivInHex* ) throw sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)

Restores a Database from an encrypted backup file.

See the Graph class EncryptedBackup method.

**Parameters**

| | |
|---|---|
| *path* | [in] Database storage file. |
| *backupFile* | [in] The Backup file to be restored. |
| *keyInHex* | [In] The AES encryption Key of the backup file as an hexadecimal string (8, 16 or 32 bytes). |
| *ivInHex* | [In] The AES Initialization Vector of the backup file as an hexadecimal string (16 bytes). |

**Returns**

A Database instance.

**Exceptions**

| | |
|---|---|
| *FileNotFoundException* | If the given file cannot be created, or the exported data file does not exists. |

#### 5.71.3.14 Database* Sparksee::RestoreEncryptedBackup ( const std::wstring & *path*, const std::wstring & *backupFile*, const SparkseeConfig & *config*, const std::wstring & *keyInHex*, const std::wstring & *ivInHex* ) throw sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)

Restores a Database from an encrypted backup file.

See the Graph class EncryptedBackup method.

**Parameters**

| | |
|---|---|
| *path* | [in] Database storage file. |
| *backupFile* | [in] The Backup file to be restored. |
| *config* | [in] Use a specific configuration instead of the one in this Sparksee |
| *keyInHex* | [In] The AES encryption Key of the backup file as an hexadecimal string (8, 16 or 32 bytes). |
| *ivInHex* | [In] The AES Initialization Vector of the backup file as an hexadecimal string (16 bytes). |

**Returns**

A Database instance.

**Exceptions**

| | |
|---|---|
| *FileNotFoundException* | If the given file cannot be created, or the exported data file does not exists. |

#### 5.71.3.15 void Sparksee::SetUnrecoverableErrorCallback ( SPARKSEE_UNRECOVERABLE_ERR_CALLBACK *callback* )

Sets the unrecoverable error callback.

**Parameters**

| | |
|---|---|
| *callback* | The unrecoverable error callback to set |

#### 5.71.3.16 bool_t Sparksee::VerifyChecksums ( const std::wstring & *path* ) throw sparksee::gdb::FileNotFoundException, sparksee::gdb::Error)

Verifies the Checksum integrity of the given image file and it's recovery log file.

The main database file checksums are always checked.

When the recovery log file exists, the recovery log file is also checked for checksum and format errors.

When the checksums of the main file or the recovery file are incorrect, the funcion returns false, in any other error it throws an exception.

Any error found is logged as a WARNING.

#### Parameters

| | |
|---|---|
| *path* | The path to the image file |

#### Returns

Returns true if the checksum verification succeeded. Returns false if an invalid checksum was detected.

#### Exceptions

| | |
|---|---|
| *sparksee::gdb::FileNotFoundException* | |
| *sparksee::gdb::Error* | |

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

- Sparksee.h

## 5.72 SparkseeConfig Class Reference

Sparksee configuration class.

#### Public Member Functions

- SparkseeConfig ()
  *Creates a new instance.*
- SparkseeConfig (const std::wstring &path)
  *Creates a new instance with a specific config file.*
- SparkseeConfig (const std::wstring &path, const std::wstring &clientId, const std::wstring &licenseId)
  *Creates a new instance with a specific config file and IDs.*
- virtual ~SparkseeConfig ()
  *Destructor.*
- int32_t GetExtentSize () const
  *Gets the size of a extent.*
- void SetExtentSize (int32_t kBytes)
  *Sets the size of the extents in KB.*
- int32_t GetExtentPages () const
  *Gets the number of pages per extent.*
- void SetExtentPages (int32_t pages)
  *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 extents)
  *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 frames)

  *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 frames)

  *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 frames)

  *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 frames)

  *Sets the maximum size for the temporary pool in number of frames.*
- int32_t GetPoolPartitions () const

  *Gets the number of partitions in each PartitionedPool.*
- void SetPoolPartitions (int32_t pools)

  *Sets the number of pools in each PartitionedPool.*
- int32_t GetCacheMaxSize () const

  *Gets the maximum size for the cache (all pools) in MB.*
- void SetCacheMaxSize (int32_t megaBytes)

  *Sets the maximum size for the cache (all pools) in MB.*
- const std::wstring & GetLicense () const

  *Gets the license key.*
- int32_t SetLicense (const std::wstring &key)

  *Sets the license key.*
- const std::wstring & GetLogFile () const

  *Gets the log file.*
- void SetLogFile (const std::wstring &filePath)

  *Sets the log file.*
- LogLevel GetLogLevel () const

  *Gets the log level.*
- void SetLogLevel (LogLevel level)

  *Sets the log level.*
- bool_t GetCacheStatisticsEnabled () const

  *Gets whether cache statistics are enabled or disabled.*
- void SetCacheStatisticsEnabled (bool_t status)

  *Enables or disables cache statistics.*
- const std::wstring & GetCacheStatisticsFile () const

  *Gets the cache statistics log file.*
- void SetCacheStatisticsFile (const std::wstring &filePath)

  *Sets the cache statistics log file.*
- int64_t GetCacheStatisticsSnapshotTime () const

  *Gets the cache statistics snapshot time in microseconds.*
- void SetCacheStatisticsSnapshotTime (int64_t microSeconds)

  *Sets the cache statistics snapshot time.*
- bool_t GetRollbackEnabled () const

  *Gets whether the rollback is enabled or disabled.*
- void SetRollbackEnabled (bool_t status)

*Enables or disables the rollback.*

- bool_t GetRecoveryEnabled () const

  *Gets whether the recovery is enabled or disabled.*
- void SetRecoveryEnabled (bool_t status)

  *Enables or disables the recovery.*
- const std::wstring & GetRecoveryLogFile () const

  *Gets the recovery log file.*
- void SetRecoveryLogFile (const std::wstring &filePath)

  *Sets the recovery log file.*
- int32_t GetRecoveryCacheMaxSize () const

  *Gets the maximum size for the recovery log cache in extents.*
- void SetRecoveryCacheMaxSize (int32_t extents)

  *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 microSeconds)

  *Sets the delay time (in microseconds) between automatic checkpoints.*
- bool_t GetChecksumEnabled () const

  *Gets whether the storage checksum usage is enabled or disabled.*
- void SetChecksumEnabled (bool_t status)

  *Enables or disables the storage checksum usage.*
- bool_t GetHighAvailabilityEnabled () const

  *Gets whether high availability mode is enabled or disabled.*
- void SetHighAvailabilityEnabled (bool_t status)

  *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 &ip)

  *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 &ip)

  *Sets the coordinators address and port list.*
- int64_t GetHighAvailabilitySynchronization () const

  *Gets the synchronization polling time.*
- void SetHighAvailabilitySynchronization (int64_t microSeconds)

  *Sets the synchronization polling time.*
- int64_t GetHighAvailabilityMasterHistory () const

  *Gets the master's history log.*
- void SetHighAvailabilityMasterHistory (int64_t filePath)

  *Sets the master's history log.*
- bool_t GetCallStackDump () const

  *Gets whether the signals will be captured to dump the call stack or not.*
- void SetCallStackDump (bool_t status)

  *Sets whether the signals will be captured to dump the call stack or not.*
- const std::wstring & GetClientId () const

  *Gets the client identifier.*
- void SetClientId (const std::wstring &clientId)

  *Set the client identifier.*
- const std::wstring & GetLicenseId () const

  *Gets the license identifier.*

- void SetLicenseId (const std::wstring &licenseId)

  *Set the license identifier.*
- int32_t GetLicensePreDownloadDays () const

  *Get the number of days before expiration when a new license will be downloaded.*
- void SetLicensePreDownloadDays (int32_t days)

  *Start trying to automatically download a new license at the specified number of days before expiration.*
- int32_t DownloadLicense ()

  *Try to download a license key Can be used to explicitly download a license when the autodownload is disabled (LicensePreDownloadDays == -1).*
- const std::wstring & GetDownloadStatus () const

  *Gets a message with the license download result.*
- bool_t DownloadExpected ()

  *Check if a new license will be automatically downloaded with the current settings.*
- std::wstring GetLicenseRequest ()

  *Get information useful to manually request a license.*
- const std::wstring & GetTmpFolder () const

  *Gets the temporary folder used for temporary staging.*
- void SetTmpFolder (const std::wstring &tmpFolder)

  *Sets the temporary folder used for temporary staging.*
- bool_t GetTmpEnabled () const

  *Gets whether using temporary storage for computations is enabled.*
- void SetTmpEnabled (bool_t enabled)

  *Sets whether to use temporary storage for computations or not.*
- int64_t GetInMemAllocSize () const

  *Gets the in-memory allocator size.*
- void SetInMemAllocSize (int64_t size)

  *Sets the in-memory allocator size.*
- bool_t Save ()

  *Save the current configuration in the specified config file.*
- bool_t SaveAll ()

  *Save all the current configuration settings in the specified config file.*
- bool_t GetEncryptionEnabled () const

  *Gets whether the storage encryption is enabled or disabled.*
- bool_t SetAESEncryptionEnabled (const std::wstring &keyInHex, const std::wstring &ivInHex)

  *Enables storage encryption using AES with the given key and iv.*
- bool_t SetAESEncryptionEnabled (int32_t keySize)

  *Enables storage encryption using AES and GENARATES a key and an iv.*
- const std::wstring & GetAESKey () const

  *Get the AES encryption key in a hexadecimal encoded string.*
- const std::wstring & GetAESIV () const

  *Get the AES initialization vector in a hexadecimal encoded string.*
- void SetEncryptionDisabled ()

  *Disables storage encryption.*
- void SetSparkseeConfigFile (const std::wstring &path)

  *Sets the config file path.*
- const std::wstring & GetSparkseeConfigFile () const

  *Gets the config file path.*

### Friends

- class **Sparksee**

### 5.72.1 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 overrided with values from a properties file (see SparkseeProperties class). Also, this settings can be overrided 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).

Checksums enabled: true ('sparksee.storage.checksum' 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).

Number of pools in the pool cluster: 0 pools ('sparksee.io.pool.clustersize' at SparkseeProperties). 0 or 1 means the clustering is disabled.

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.

Capture abort signals to dump the call stack ('sparksee.callstackdump' at SparkseeProperties) is enabled by default on most platforms.

**Author**

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

#### 5.72.2 Constructor & Destructor Documentation

##### 5.72.2.1 SparkseeConfig::SparkseeConfig ( )

Creates a new instance.

Values are set with default values.

##### 5.72.2.2 SparkseeConfig::SparkseeConfig ( const std::wstring & *path* )

Creates a new instance with a specific config file.

The config file will be loaded using the global properties class and the file may be automatically overwritten with the config changes. If the file doesn't exists, it will be created.

**Parameters**

| | |
|---|---|
| *path* | [in] File path to the config file. |

##### 5.72.2.3 SparkseeConfig::SparkseeConfig ( const std::wstring & *path*, const std::wstring & *clientId*, const std::wstring & *licenseId* )

Creates a new instance with a specific config file and IDs.

The config file will be loaded using the global properties class and the file may be automatically overwritten with the config changes. If the config file already exists, the client and license ids should have the same values as the given arguments. If the file doesn't exists, it will be created.

**Parameters**

| | |
|---|---|
| *path* | [in] File path to the config file. |
| *clientId* | [in] The client identifier. |
| *licenseId* | [in] The license identifier. |

#### 5.72.3 Member Function Documentation

##### 5.72.3.1 int32_t SparkseeConfig::DownloadLicense ( )

Try to download a license key Can be used to explicitly download a license when the autodownload is disabled (LicensePreDownloadDays == -1).

**Returns**

Returns -1 if a valid license key couldn't be downloaded, 0 if the same license was downloaded or 1 if a different license is downloaded.

#### 5.72.3.2 const std::wstring& SparkseeConfig::GetAESIV ( ) const

Get the AES initialization vector in a hexadecimal encoded string.

**Returns**

The AES initialization vector as an hexadecimal string.

#### 5.72.3.3 const std::wstring& SparkseeConfig::GetAESKey ( ) const

Get the AES encryption key in a hexadecimal encoded string.

**Returns**

The AES encryption Key as an hexadecimal string.

#### 5.72.3.4 int32_t SparkseeConfig::GetCacheMaxSize ( ) const

Gets the maximum size for the cache (all pools) in MB.

**Returns**

The maximum size for the cache (all pools) in MB.

#### 5.72.3.5 bool_t SparkseeConfig::GetCacheStatisticsEnabled ( ) const

Gets whether cache statistics are enabled or disabled.

**Returns**

TRUE if cache statistics are enabled, FALSE otherwise.

#### 5.72.3.6 const std::wstring& SparkseeConfig::GetCacheStatisticsFile ( ) const

Gets the cache statistics log file.

Useless if cache statistics are disabled.

**Returns**

The cache statistics log file.

#### 5.72.3.7 int64_t SparkseeConfig::GetCacheStatisticsSnapshotTime ( ) const

Gets the cache statistics snapshot time in microseconds.

Useless if cache statistics are disabled.

**Returns**

The cache statistics snapshot time in microseconds.

#### 5.72.3.8 bool_t SparkseeConfig::GetCallStackDump ( ) const

Gets whether the signals will be captured to dump the call stack or not.

**Returns**

TRUE if the signals must be captured, FALSE otherwise.

#### 5.72.3.9 bool_t SparkseeConfig::GetChecksumEnabled ( ) const

Gets whether the storage checksum usage is enabled or disabled.

**Returns**

TRUE if the checksum is enabled, FALSE otherwise.

#### 5.72.3.10 const std::wstring& SparkseeConfig::GetClientId ( ) const

Gets the client identifier.

**Returns**

The client identifier.

#### 5.72.3.11 const std::wstring& SparkseeConfig::GetDownloadStatus ( ) const

Gets a message with the license download result.

It can be used when the DownloadLicense call failed.

**Returns**

The log download result.

#### 5.72.3.12 bool_t SparkseeConfig::GetEncryptionEnabled ( ) const

Gets whether the storage encryption is enabled or disabled.

**Returns**

TRUE if the encryption is enabled, FALSE otherwise.

#### 5.72.3.13 int32_t SparkseeConfig::GetExtentPages ( ) const

Gets the number of pages per extent.

**Returns**

The number of pages per extent.

#### 5.72.3.14 int32_t SparkseeConfig::GetExtentSize ( ) const

Gets the size of a extent.

**Returns**

The size of a extent in KB.

#### 5.72.3.15 const std::wstring& SparkseeConfig::GetHighAvailabilityCoordinators ( ) const

Gets the coordinators address and port list.

**Returns**

The coordinators address and port list.

#### 5.72.3.16 bool_t SparkseeConfig::GetHighAvailabilityEnabled ( ) const

Gets whether high availability mode is enabled or disabled.

**Returns**

TRUE if high availability mode is enabled, FALSE otherwise.

#### 5.72.3.17 const std::wstring& SparkseeConfig::GetHighAvailabilityIP ( ) const

Gets the IP address and port of the instance.

**Returns**

The IP address and port of the instance.

#### 5.72.3.18 int64_t SparkseeConfig::GetHighAvailabilityMasterHistory ( ) const

Gets the master's history log.

**Returns**

The master's history log.

#### 5.72.3.19 int64_t SparkseeConfig::GetHighAvailabilitySynchronization ( ) const

Gets the synchronization polling time.

**Returns**

The Synchronization polling time.

#### 5.72.3.20 int64_t SparkseeConfig::GetInMemAllocSize ( ) const

Gets the in-memory allocator size.

**Returns**

Returns the in-memory allocator size

#### 5.72.3.21 const std::wstring& SparkseeConfig::GetLicense ( ) const

Gets the license key.

**Returns**

The license key.

#### 5.72.3.22 const std::wstring& SparkseeConfig::GetLicenseId ( ) const

Gets the license identifier.

**Returns**

The license identifier.

#### 5.72.3.23 int32_t SparkseeConfig::GetLicensePreDownloadDays ( ) const

Get the number of days before expiration when a new license will be downloaded.

**Returns**

Returns the number of days or -1 if it will never be automatically downloaded.

#### 5.72.3.24 const std::wstring& SparkseeConfig::GetLogFile ( ) const

Gets the log file.

**Returns**

The log file.

#### 5.72.3.25 LogLevel SparkseeConfig::GetLogLevel ( ) const

Gets the log level.

**Returns**

The LogLevel.

#### 5.72.3.26 int32_t SparkseeConfig::GetPoolFrameSize ( ) const

Gets the size of a pool frame in number of extents.

**Returns**

The size of a pool frame in number of extents.

#### 5.72.3.27 int32_t SparkseeConfig::GetPoolPartitions ( ) const

Gets the number of partitions in each PartitionedPool.

**Returns**

The number of partitions in each PartitionedPool.

#### 5.72.3.28 int32_t SparkseeConfig::GetPoolPersistentMaxSize ( ) const

Gets the maximum size for the persistent pool in number of frames.

**Returns**

The maximum size for the persistent pool in number of frames.

#### 5.72.3.29 int32_t SparkseeConfig::GetPoolPersistentMinSize ( ) const

Gets the minimum size for the persistent pool in number of frames.

**Returns**

The minimum size for the persistent pool in number of frames.

#### 5.72.3.30 int32_t SparkseeConfig::GetPoolTemporaryMaxSize ( ) const

Gets the maximum size for the temporary pool in number of frames.

**Returns**

The maximum size for the temporary pool in number of frames.

#### 5.72.3.31 int32_t SparkseeConfig::GetPoolTemporaryMinSize ( ) const

Gets the minimum size for the temporary pool in number of frames.

**Returns**

The minimum size for the temporary pool in number of frames.

#### 5.72.3.32 int32_t SparkseeConfig::GetRecoveryCacheMaxSize ( ) const

Gets the maximum size for the recovery log cache in extents.

**Returns**

The maximum size for the recovery log cache in extents.

#### 5.72.3.33 int64_t SparkseeConfig::GetRecoveryCheckpointTime ( ) const

Gets the delay time (in microseconds) between automatic checkpoints.

**Returns**

The delay time (in microseconds) between automatic checkpoints.

#### 5.72.3.34 bool_t SparkseeConfig::GetRecoveryEnabled ( ) const

Gets whether the recovery is enabled or disabled.

**Returns**

TRUE if the recovery is enabled, FALSE otherwise.

#### 5.72.3.35 const std::wstring& SparkseeConfig::GetRecoveryLogFile ( ) const

Gets the recovery log file.

**Returns**

The recovery log file.

#### 5.72.3.36 bool_t SparkseeConfig::GetRollbackEnabled ( ) const

Gets whether the rollback is enabled or disabled.

**Returns**

TRUE if the rollback is enabled, FALSE otherwise.

#### 5.72.3.37 const std::wstring& SparkseeConfig::GetSparkseeConfigFile ( ) const

Gets the config file path.

**Returns**

Returns the configured path or an empty string

#### 5.72.3.38 bool_t SparkseeConfig::GetTmpEnabled ( ) const

Gets whether using temporary storage for computations is enabled.

**Returns**

True if enabled

#### 5.72.3.39 const std::wstring& SparkseeConfig::GetTmpFolder ( ) const

Gets the temporary folder used for temporary staging.

**Returns**

The temporary folder path

#### 5.72.3.40 bool_t SparkseeConfig::Save ( )

Save the current configuration in the specified config file.

It will try to save the current configuration only if a config file was specified. This method will be automatically used when a Sparksee class downloads a new License.

**Returns**

Returns true if the config file is successfully written or false otherwise.

#### 5.72.3.41 bool_t SparkseeConfig::SaveAll ( )

Save all the current configuration settings in the specified config file.

It will try to save the current configuration only if a config file was specified. The saved config file WILL CONATAIN all the modified settings including the secret ENCRYPTION KEYS. You should usually just use the Save method instead.

**Returns**

Returns true if the config file is successfully written or false otherwise.

#### 5.72.3.42 bool_t SparkseeConfig::SetAESEncryptionEnabled ( const std::wstring & *keyInHex*, const std::wstring & *ivInHex* )

Enables storage encryption using AES with the given key and iv.

The key and initialization vector will not be saved in the config file.

**Parameters**

| | |
|---|---|
| *keyInHex* | [In] The AES encryption Key as an hexadecimal string (8, 16 or 32 bytes). |
| *ivInHex* | [In] The AES Initialization Vector as an hexadecimal string (16 bytes). |

**Returns**

Returns true if the encryption had been enabled with the given arguments or false otherwise.

#### 5.72.3.43 bool_t SparkseeConfig::SetAESEncryptionEnabled ( int32_t *keySize* )

Enables storage encryption using AES and GENARATES a key and an iv.

YOU MUST GET AND KEEP SAFE THE GENERATED KEY AND IV! The key and initialization vector will not be saved in the config file.

**Parameters**

| | |
|---|---|
| *keySize* | [In] The key size in bytes (8, 16 or 32). |

**Returns**

Returns true if the encryption had been enabled.

#### 5.72.3.44 void SparkseeConfig::SetCacheMaxSize ( int32_t *megaBytes* )

Sets the maximum size for the cache (all pools) in MB.

**Parameters**

| | |
|---|---|
| *megaBytes* | [in] The maximum size for the cache (all pools) in MB. It must be non-negative. |

#### 5.72.3.45 void SparkseeConfig::SetCacheStatisticsEnabled ( bool_t *status* )

Enables or disables cache statistics.

**Parameters**

| | |
|---|---|
| *status* | [in] If TRUE this enables cache statistics, if FALSE this disables cache statistics. |

#### 5.72.3.46 void SparkseeConfig::SetCacheStatisticsFile ( const std::wstring & *filePath* )

Sets the cache statistics log file.

Useless if cache statistics are disabled.

**Parameters**

| | |
|---|---|
| *filePath* | [in] The cache statistics log file. |

#### 5.72.3.47 void SparkseeConfig::SetCacheStatisticsSnapshotTime ( int64_t *microSeconds* )

Sets the cache statistics snapshot time.

Useless if cache statistics are disabled.

**Parameters**

| | |
|---|---|
| *microSeconds* | [in] The cache statistics snapshot time in microseconds. |

#### 5.72.3.48 void SparkseeConfig::SetCallStackDump ( bool_t *status* )

Sets whether the signals will be captured to dump the call stack or not.

**Parameters**

| | |
|---|---|
| *status* | [in] If TRUE signals must be captured. |

#### 5.72.3.49 void SparkseeConfig::SetChecksumEnabled ( bool_t *status* )

Enables or disables the storage checksum usage.

**Parameters**

| | |
|---|---|
| *status* | [in] If TRUE this enables the checksum, if FALSE then disables it. |

#### 5.72.3.50 void SparkseeConfig::SetClientId ( const std::wstring & *clientId* )

Set the client identifier.

If you don't have a client identifier, you may want to register at http://sparsity-technologies.↩com/#sparksee

**Parameters**

| | |
|---|---|
| *client↩Id* | [in] The client identifier. |

#### 5.72.3.51 void SparkseeConfig::SetExtentPages ( int32_t *pages* )

Sets the number of pages per extent.

**Parameters**

| | |
|---|---|
| *pages* | [in] The number of pages. It must be at least 1 page and the page size must be greater than or equal to 4KB. |

#### 5.72.3.52 void SparkseeConfig::SetExtentSize ( int32_t *kBytes* )

Sets the size of the extents in KB.

**Parameters**

| | |
|---|---|
| *kBytes* | [in] The size of an extent in KB. An extent can have a size between 4KB and 64KB, and it must be a power of 2. |

#### 5.72.3.53 void SparkseeConfig::SetHighAvailabilityCoordinators ( const std::wstring & *ip* )

Sets the coordinators address and port list.

**Parameters**

| | |
|---|---|
| *ip* | [in] The coordinators address and port list. |

#### 5.72.3.54 void SparkseeConfig::SetHighAvailabilityEnabled ( bool_t *status* )

Enables or disables high availability mode.

**Parameters**

| | |
|---|---|
| *status* | [in] If TRUE this enables high availability mode, if FALSE this disables high availability mode. |

#### 5.72.3.55 void SparkseeConfig::SetHighAvailabilityIP ( const std::wstring & *ip* )

Sets the IP address and port of the instance.

**Parameters**

| | |
|---|---|
| *ip* | [in] The IP address and port of the instance. |

#### 5.72.3.56 void SparkseeConfig::SetHighAvailabilityMasterHistory ( int64_t *filePath* )

Sets the master's history log.

**Parameters**

| | |
|---|---|
| *filePath* | [in] The master's history log. |

#### 5.72.3.57 void SparkseeConfig::SetHighAvailabilitySynchronization ( int64_t *microSeconds* )

Sets the synchronization polling time.

**Parameters**

| | |
|---|---|
| *microSeconds* | [in] The synchronization polling time. |

#### 5.72.3.58 void SparkseeConfig::SetInMemAllocSize ( int64_t *size* )

Sets the in-memory allocator size.

**Parameters**

| | |
|---|---|
| *size* | The size to set the in-memory allocator to |

#### 5.72.3.59 int32_t SparkseeConfig::SetLicense ( const std::wstring & *key* )

Sets the license key.

**Parameters**

| | |
|---|---|
| *key* | [in] The license key. Returns -1 if the new key can't be used, 0 if it's the same license or 1 if the new license is applied. |

#### 5.72.3.60 void SparkseeConfig::SetLicenseId ( const std::wstring & *licenseId* )

Set the license identifier.

If you don't have a license identifier, you may want to register at http://sparsity-technologies.↩com/#sparksee

**Parameters**

| | |
|---|---|
| *license*↩*Id* | [in] The license identifier. |

#### 5.72.3.61 void SparkseeConfig::SetLicensePreDownloadDays ( int32_t *days* )

Start trying to automatically download a new license at the specified number of days before expiration.

**Parameters**

| | |
|---|---|
| *days* | [in] Number of days before expiration or -1 if the license should never be downloaded. |

#### 5.72.3.62 void SparkseeConfig::SetLogFile ( const std::wstring & *filePath* )

Sets the log file.

**Parameters**

| | |
|---|---|
| *filePath* | [in] The log file. |

#### 5.72.3.63 void SparkseeConfig::SetLogLevel ( LogLevel *level* )

Sets the log level.

**Parameters**

| | |
|---|---|
| *level* | [in] The LogLevel. |

#### 5.72.3.64 void SparkseeConfig::SetPoolFrameSize ( int32_t *extents* )

Sets the size of a pool frame in number of extents.

**Parameters**

| | |
|---|---|
| *extents* | [in] The size of a pool frame in number of extents. It must be non-negative. |

#### 5.72.3.65 void SparkseeConfig::SetPoolPartitions ( int32_t *pools* )

Sets the number of pools in each PartitionedPool.

**Parameters**

| | |
|---|---|
| *pools* | [in] The number of pools in each PartitionedPool. It must be non-negative. |

#### 5.72.3.66 void SparkseeConfig::SetPoolPersistentMaxSize ( int32_t *frames* )

Sets the maximum size for the persistent pool in number of frames.

**Parameters**

| | |
|---|---|
| *frames* | [in] The maximum size for the persistent pool in number of frames. It must be non-negative. |

#### 5.72.3.67 void SparkseeConfig::SetPoolPersistentMinSize ( int32_t *frames* )

Sets the minimum size for the persistent pool in number of frames.

**Parameters**

| | |
|---|---|
| *frames* | [in] The minimum size for the persistent pool in number of frames. It must be non-negative. |

#### 5.72.3.68 void SparkseeConfig::SetPoolTemporaryMaxSize ( int32_t *frames* )

Sets the maximum size for the temporary pool in number of frames.

**Parameters**

| | |
|---|---|
| *frames* | [in] The maximum size for the temporary pool in number of frames. It must be non-negative. |

#### 5.72.3.69 void SparkseeConfig::SetPoolTemporaryMinSize ( int32_t *frames* )

Sets the minimum size for the temporary pool in number of frames.

**Parameters**

| | |
|---|---|
| *frames* | [in] The minimum size for the temporary pool in number of frames. It must be non-negative. |

#### 5.72.3.70 void SparkseeConfig::SetRecoveryCacheMaxSize ( int32_t *extents* )

Sets the maximum size for the recovery log cache in extents.

**Parameters**

| | |
|---|---|
| *extents* | [in] The maximum size for the recovery log cache in extents. A 0 sets the default value (extents up to 1MB). |

#### 5.72.3.71 void SparkseeConfig::SetRecoveryCheckpointTime ( int64_t *microSeconds* )

Sets the delay time (in microseconds) between automatic checkpoints.

**Parameters**

| | |
|---|---|
| *microSeconds* | [in] The delay time (in microseconds) between automatic checkpoints. A 0 forces a checkpoint after each commited transaction. |

#### 5.72.3.72 void SparkseeConfig::SetRecoveryEnabled ( bool_t *status* )

Enables or disables the recovery.

**Parameters**

| | |
|---|---|
| *status* | [in] If TRUE this enables the recovery, if FALSE then disables it. |

#### 5.72.3.73 void SparkseeConfig::SetRecoveryLogFile ( const std::wstring & *filePath* )

Sets the recovery log file.

**Parameters**

| | |
|---|---|
| *filePath* | [in] The recovery log file. Left it empty for the default log file (same as <database_file_name>.log) |

#### 5.72.3.74 void SparkseeConfig::SetRollbackEnabled ( bool_t *status* )

Enables or disables the rollback.

**Parameters**

| | |
|---|---|
| *status* | [in] If TRUE this enables the rollback, if FALSE then disables it. |

#### 5.72.3.75 void SparkseeConfig::SetSparkseeConfigFile ( const std::wstring & *path* )

Sets the config file path.

The file is not loaded in this operation, see the constructor options if you need to load the file. The config file may be automatically overwritten with the config changes. If the file doesn't exists, it will be created.

**Parameters**

| | |
|---|---|
| *path* | [in] File path to the config file. |

#### 5.72.3.76 void SparkseeConfig::SetTmpEnabled ( bool_t *enabled* )

Sets whether to use temporary storage for computations or not.

**Parameters**

| | |
|---|---|
| *enabled* | True to use temporary storage for computation |

#### 5.72.3.77 void SparkseeConfig::SetTmpFolder ( const std::wstring & *tmpFolder* )

Sets the temporary folder used for temporary staging.

**Parameters**

| | |
|---|---|
| *tmpFolder* | The temporary folder to set |

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

- SparkseeConfig.h

## 5.73 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.*

- static void Clear ()

  *Remove all the properties.*

- static void SetHI (const std::wstring &value)

  *YOU SHOULD NOT USE THIS METHOD.*

#### Friends

- class **SparkseeConfig**

### 5.73.1 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.

### 5.73.2 Member Function Documentation

#### 5.73.2.1 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.

#### 5.73.2.2 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.

#### 5.73.2.3 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.

#### 5.73.2.4 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.

#### 5.73.2.5 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. |

#### 5.73.2.6 static void SparkseeProperties::SetHl ( const std::wstring & *value* ) [static]

YOU SHOULD NOT USE THIS METHOD.

This method exists only for a specific service.

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

- SparkseeConfig.h

## 5.74 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.*

### 5.74.1 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

### 5.74.2 Constructor & Destructor Documentation

#### 5.74.2.1 StringList::StringList ( )

Constructor.

This creates an empty list.

#### 5.74.2.2 StringList::StringList ( const std::vector< std::wstring > & *v* )

Constructor.

**Parameters**

| | |
|---|---|
| *v* | [in] Vector. |

#### 5.74.3 Member Function Documentation

##### 5.74.3.1 void StringList::Add ( const std::wstring & *str* ) [inline]

Adds a String at the end of the list.

**Parameters**

| | |
|---|---|
| *str* | [in] String. |

##### 5.74.3.2 int32_t StringList::Count ( ) const [inline]

Number of elements in the list.

**Returns**

Number of elements in the list.

##### 5.74.3.3 StringListIterator* StringList::Iterator ( )

Gets a new StringListIterator.

**Returns**

StringListIterator instance.

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

- Graph_data.h

## 5.75 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**

### 5.75.1 Detailed Description

StringList iterator class.

Iterator to traverse all the strings into a StringList instance.

#### Author

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

### 5.75.2 Member Function Documentation

#### 5.75.2.1 bool_t StringListIterator::HasNext ( ) [inline]

Gets if there are more elements.

#### Returns

TRUE if there are more elements, FALSE otherwise.

#### 5.75.2.2 const std::wstring& StringListIterator::Next ( ) [inline]

Moves to the next element.

#### Returns

The next element.

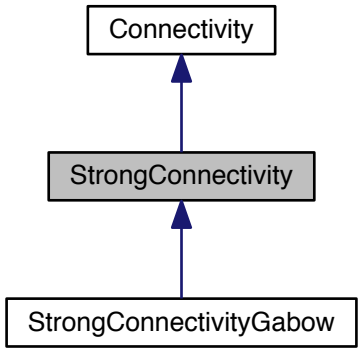
The documentation for this class was generated from the following file:

- Graph_data.h

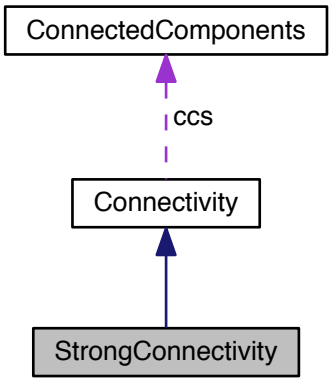
## 5.76 StrongConnectivity Class Reference

StrongConnectivity class.

Inheritance diagram for StrongConnectivity:




Collaboration diagram for StrongConnectivity:




### Public Member Functions

- virtual ~StrongConnectivity ()

  *Destructor.*
- virtual void AddEdgeType (sparksee::gdb::type_t type, sparksee::gdb::EdgesDirection dir)

  *Allows connectivity through edges of the given type.*
- virtual void AddAllEdgeTypes (sparksee::gdb::EdgesDirection dir)

*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.*

### 5.76.1 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

### 5.76.2 Constructor & Destructor Documentation

#### 5.76.2.1 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 |

### 5.76.3 Member Function Documentation

#### 5.76.3.1 virtual void StrongConnectivity::AddAllEdgeTypes ( sparksee::gdb::EdgesDirection *dir* ) [virtual]

Allows connectivity through all edge types of the graph.

**Parameters**

| | |
|---|---|
| *dir* | [in] Edge direction. |

#### 5.76.3.2 virtual void StrongConnectivity::AddEdgeType ( sparksee::gdb::type_t *type*, sparksee::gdb::EdgesDirection *dir* ) [virtual]

Allows connectivity through edges of the given type.

**Parameters**

| | |
|---|---|
| *type* | [in] Edge type. |
| *dir* | [in] Edge direction. |

#### 5.76.3.3 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. |

#### 5.76.3.4 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. |

#### 5.76.3.5 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 WeakConnectivityDFS, and StrongConnectivityGabow.

#### 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::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.

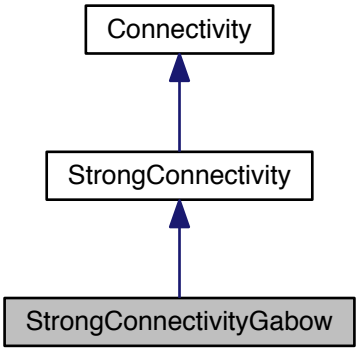
The documentation for this class was generated from the following file:

- StrongConnectivity.h

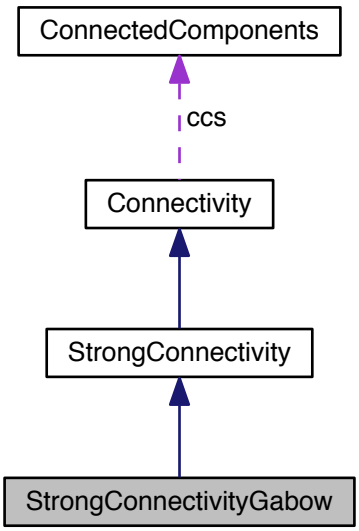
## 5.77 StrongConnectivityGabow Class Reference

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 &session)

  *Creates a new instance of StrongConnectivityGabow.*
- virtual ~StrongConnectivityGabow ()

  *Destructor.*
- void Run ()

  *Executes the algorithm.*
- virtual void AddEdgeType (sparksee::gdb::type_t type, sparksee::gdb::EdgesDirection dir)

  *Allows connectivity through edges of the given type.*
- virtual void AddAllEdgeTypes (sparksee::gdb::EdgesDirection dir)

  *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.*

### 5.77.1 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

### 5.77.2 Constructor & Destructor Documentation

#### 5.77.2.1 StrongConnectivityGabow::StrongConnectivityGabow ( sparksee::gdb::Session & *session* )

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**

| | |
|---|---|
| *session* | [in] Session to get the graph from and calculate the connectivity |

### 5.77.3 Member Function Documentation

#### 5.77.3.1 virtual void StrongConnectivity::AddAllEdgeTypes ( sparksee::gdb::EdgesDirection *dir* ) [virtual], [inherited]

Allows connectivity through all edge types of the graph.

**Parameters**

| | |
|---|---|
| *dir* | [in] Edge direction. |

#### 5.77.3.2 virtual void StrongConnectivity::AddEdgeType ( sparksee::gdb::type_t *type*, sparksee::gdb::EdgesDirection *dir* ) [virtual], [inherited]

Allows connectivity through edges of the given type.

**Parameters**

| | |
|---|---|
| *type* | [in] Edge type. |
| *dir* | [in] Edge direction. |

#### 5.77.3.3 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. |

#### 5.77.3.4 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. |

#### 5.77.3.5 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.

#### 5.77.3.6 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. |

#### 5.77.3.7 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.

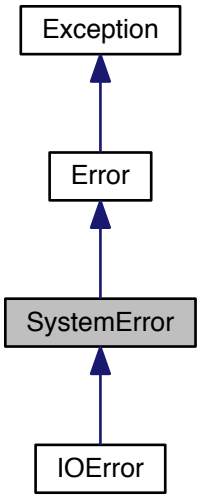
The documentation for this class was generated from the following file:

- StrongConnectivityGabow.h

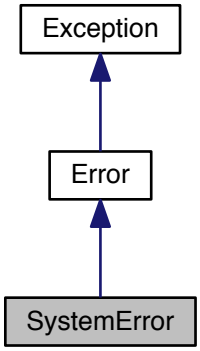
## 5.78 SystemError Class Reference

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 void ThrowError (int32_t coreErrorCode)
  *Throws a new Error instance from a sparksee_core error code.*

#### Protected Attributes

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

### 5.78.1 Detailed Description

System error class.

#### Author

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

### 5.78.2 Constructor & Destructor Documentation

#### 5.78.2.1 SystemError::SystemError ( 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 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.

#### 5.78.3.2 void Exception::SetMessage ( const std::string & *mess* ) [inherited]

Sets the message of the exception.

**Parameters**

| | |
|---|---|
| *mess* | [in] Message. |

#### 5.78.3.3 static void Error::ThrowError ( int32_t *coreErrorCode* ) [static], [inherited]

Throws 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 will throw an Error instance of an specific Error subclass instance.

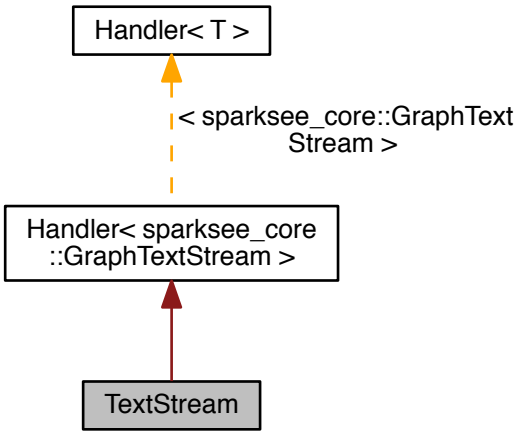
The documentation for this class was generated from the following file:

- Exception.h

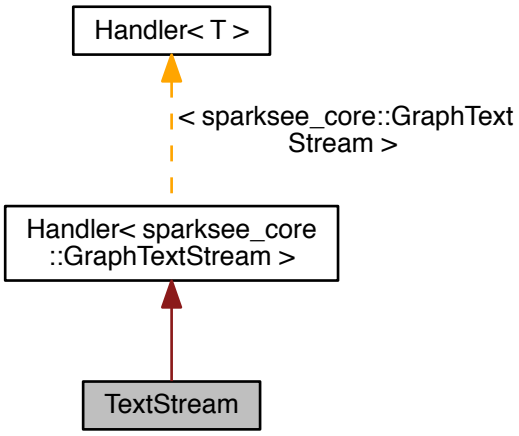
## 5.79 TextStream Class Reference

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.*

#### Private Member Functions

- sparksee_core::GraphTextStream * GetHandler ()

  *Gets the handled reference.*

- const sparksee_core::GraphTextStream * GetHandler () const

  *Gets the handled reference.*

- void SetHandler (sparksee_core::GraphTextStream *h)

  *Sets the handled reference.*

- void FreeHandler ()

  *Frees (deletes) the handled reference.*

#### Friends

- class **Graph**

### 5.79.1 Detailed Description

TextStream class.

It allows for reading and writting Text attribute values.

It is very important to close the stream once no more reading or writting 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) exeucte 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 Get↩ AttributeText 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

### 5.79.2 Constructor & Destructor Documentation

#### 5.79.2.1 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 begining of the stream. |

### 5.79.3 Member Function Documentation

#### 5.79.3.1 void TextStream::Close ( )

Closes the stream.

Once the Stream is closed, it cannot be used again.

Closing the stream is mandatory when the stream is not null and strongly recommended when it's null to avoid deallocation problems in some platforms.

#### 5.79.3.2 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

#### 5.79.3.3 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 ($<=$ length). If 0, there is no more data to be read from the stream.

#### 5.79.3.4 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 $> 0$. |

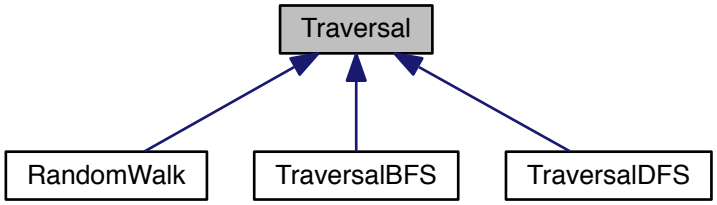
The documentation for this class was generated from the following file:

- Stream.h

## 5.80 Traversal Class Reference

Traversal class.

Inheritance diagram for Traversal:




### Public Member Functions

- virtual void AddEdgeType (sparksee::gdb::type_t type, sparksee::gdb::EdgesDirection dir)
  *Allows for traversing edges of the given type.*
- virtual void AddAllEdgeTypes (sparksee::gdb::EdgesDirection dir)
  *Allows for traversing all edge types of the graph.*
- virtual void AddNodeType (sparksee::gdb::type_t type)
  *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.*

### 5.80.1 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

### 5.80.2 Constructor & Destructor Documentation

#### 5.80.2.1 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. |

### 5.80.3 Member Function Documentation

#### 5.80.3.1 virtual void Traversal::AddAllEdgeTypes ( sparksee::gdb::EdgesDirection *dir* ) [virtual]

Allows for traversing all edge types of the graph.

**Parameters**

| | |
|---|---|
| *dir* | [in] Edge direction. |

Reimplemented in RandomWalk.

#### 5.80.3.2 virtual void Traversal::AddEdgeType ( sparksee::gdb::type_t *type*, sparksee::gdb::EdgesDirection *dir* ) [virtual]

Allows for traversing edges of the given type.

**Parameters**

| | |
|---|---|
| *type* | [in] Edge type. |
| *dir* | [in] Edge direction. |

Reimplemented in RandomWalk.

#### 5.80.3.3 virtual void Traversal::AddNodeType ( sparksee::gdb::type_t *type* ) [virtual]

Allows for traversing nodes of the given type.

**Parameters**

| | |
|---|---|
| *type* | The node type to add |

Reimplemented in RandomWalk.

#### 5.80.3.4 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. |

Reimplemented in RandomWalk.

#### 5.80.3.5 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. |

Reimplemented in RandomWalk.

#### 5.80.3.6 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 RandomWalk, TraversalBFS, and TraversalDFS.

#### 5.80.3.7 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 RandomWalk, TraversalBFS, and TraversalDFS.

#### 5.80.3.8 virtual sparksee::gdb::oid_t Traversal::Next ( ) [pure virtual]

Gets the next object of the traversal.

**Returns**

A node or edge identifier.

Implemented in RandomWalk, TraversalBFS, and TraversalDFS.

#### 5.80.3.9 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. |

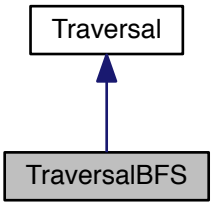
The documentation for this class was generated from the following file:

- Traversal.h

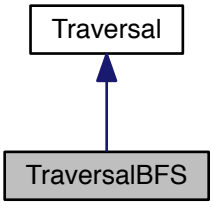
## 5.81 TraversalBFS Class Reference

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 &session, sparksee::gdb::oid_t node)

  *Creates a new instance.*
- virtual void AddEdgeType (sparksee::gdb::type_t type, sparksee::gdb::EdgesDirection dir)

  *Allows for traversing edges of the given type.*
- virtual void AddAllEdgeTypes (sparksee::gdb::EdgesDirection dir)

  *Allows for traversing all edge types of the graph.*
- virtual void AddNodeType (sparksee::gdb::type_t type)

  *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.*

### 5.81.1 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

### 5.81.2 Constructor & Destructor Documentation

#### 5.81.2.1 TraversalBFS::TraversalBFS ( sparksee::gdb::Session & *session*, sparksee::gdb::oid_t *node* )

Creates a new instance.

**Parameters**

| | |
|---|---|
| *session* | [in] Session to get the graph from and perform traversal. |
| *node* | [in] Node to start traversal from. |

### 5.81.3 Member Function Documentation

#### 5.81.3.1 virtual void Traversal::AddAllEdgeTypes ( sparksee::gdb::EdgesDirection *dir* ) [virtual], [inherited]

Allows for traversing all edge types of the graph.

**Parameters**

| | |
|---|---|
| *dir* | [in] Edge direction. |

Reimplemented in RandomWalk.

#### 5.81.3.2 virtual void Traversal::AddEdgeType ( sparksee::gdb::type_t *type*, sparksee::gdb::EdgesDirection *dir* ) [virtual], [inherited]

Allows for traversing edges of the given type.

**Parameters**

| | |
|---|---|
| *type* | [in] Edge type. |
| *dir* | [in] Edge direction. |

Reimplemented in RandomWalk.

#### 5.81.3.3 virtual void Traversal::AddNodeType ( sparksee::gdb::type_t *type* ) [virtual], [inherited]

Allows for traversing nodes of the given type.

**Parameters**

| | |
|---|---|
| *type* | The node type to add |

Reimplemented in RandomWalk.

#### 5.81.3.4 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. |

Reimplemented in RandomWalk.

#### 5.81.3.5 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. |

Reimplemented in RandomWalk.

#### 5.81.3.6 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.

#### 5.81.3.7 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.

#### 5.81.3.8 virtual sparksee::gdb::oid_t TraversalBFS::Next ( ) [virtual]

Gets the next object of the traversal.

**Returns**

A node or edge identifier.

Implements Traversal.

#### 5.81.3.9 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. |

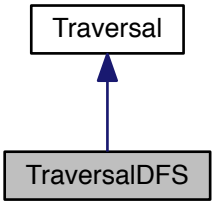
The documentation for this class was generated from the following file:

- TraversalBFS.h

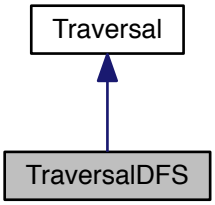
## 5.82 TraversalDFS Class Reference

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 &session, sparksee::gdb::oid_t node)

*Creates a new instance.*

- virtual void AddEdgeType (sparksee::gdb::type_t type, sparksee::gdb::EdgesDirection dir)

  *Allows for traversing edges of the given type.*
- virtual void AddAllEdgeTypes (sparksee::gdb::EdgesDirection dir)

  *Allows for traversing all edge types of the graph.*
- virtual void AddNodeType (sparksee::gdb::type_t type)

  *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.*

### 5.82.1 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

### 5.82.2 Constructor & Destructor Documentation

#### 5.82.2.1 TraversalDFS::TraversalDFS ( sparksee::gdb::Session & *session*, sparksee::gdb::oid_t *node* )

Creates a new instance.

**Parameters**

| | |
|---|---|
| *session* | [in] Session to get the graph from and perform traversal. |
| *node* | [in] Node to start traversal from. |

### 5.82.3 Member Function Documentation

#### 5.82.3.1 virtual void Traversal::AddAllEdgeTypes ( sparksee::gdb::EdgesDirection *dir* ) [virtual], [inherited]

Allows for traversing all edge types of the graph.

**Parameters**

| | |
|---|---|
| *dir* | [in] Edge direction. |

Reimplemented in RandomWalk.

#### 5.82.3.2 virtual void Traversal::AddEdgeType ( sparksee::gdb::type_t *type*, sparksee::gdb::EdgesDirection *dir* ) [virtual], [inherited]

Allows for traversing edges of the given type.

**Parameters**

| | |
|---|---|
| *type* | [in] Edge type. |
| *dir* | [in] Edge direction. |

Reimplemented in RandomWalk.

#### 5.82.3.3 virtual void Traversal::AddNodeType ( sparksee::gdb::type_t *type* ) [virtual], [inherited]

Allows for traversing nodes of the given type.

**Parameters**

| | |
|---|---|
| *type* | The node type to add |

Reimplemented in RandomWalk.

#### 5.82.3.4 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. |

Reimplemented in RandomWalk.

#### 5.82.3.5 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. |

Reimplemented in RandomWalk.

#### 5.82.3.6 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.

#### 5.82.3.7 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.

#### 5.82.3.8 virtual sparksee::gdb::oid_t TraversalDFS::Next ( ) [virtual]

Gets the next object of the traversal.

**Returns**

A node or edge identifier.

Implements Traversal.

#### 5.82.3.9 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

## 5.83 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.*

- static const type_t GlobalType

  *Global type identifier.*

- static const type_t NodesType

  *Identifier for all nodeType attributes.*

- static const type_t EdgesType

  *Identifier for all edgeType attributes.*

#### Friends

- class **Graph**

### 5.83.1 Detailed Description

Type data class.

It contains information about a node or edge type.

#### Author

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

### 5.83.2 Member Function Documentation

#### 5.83.2.1 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.

#### 5.83.2.2 type_t Type::GetId ( ) const [inline]

Gets the Sparksee type identifier.

**Returns**

The Sparksee type identifier.

#### 5.83.2.3 bool_t Type::GetIsDirected ( ) const [inline]

Gets if this is a directed edge type.

**Returns**

TRUE for directed edge types, FALSE otherwise.

#### 5.83.2.4 bool_t Type::GetIsRestricted ( ) const [inline]

Gets if this is a restricted edge type.

**Returns**

TRUE for restricted edge types, FALSE otherwise.

#### 5.83.2.5 const std::wstring& Type::GetName ( ) const [inline]

Gets the unique type name.

**Returns**

The unique type name.

#### 5.83.2.6 int64_t Type::GetNumObjects ( ) const [inline]

Gets the number of objects belonging to the type.

**Returns**

The number of objects belonging to the type.

#### 5.83.2.7 ObjectType Type::GetObjectType ( ) const [inline]

Gets the object type.

**Returns**

The object type.

#### 5.83.2.8 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.

#### 5.83.2.9 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.

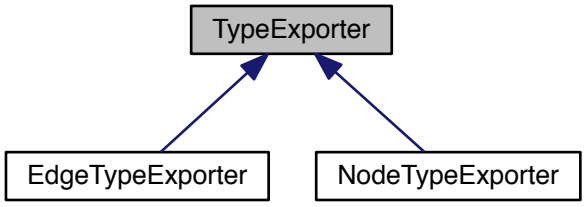
The documentation for this class was generated from the following file:

- Graph_data.h

## 5.84 TypeExporter Class Reference

Base TypeExporter class.

Inheritance 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 &graph)

  *Sets the graph that will be exported.*
- void SetType (sparksee::gdb::type_t type)

  *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.*

### 5.84.1 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

### 5.84.2 Constructor & Destructor Documentation

#### 5.84.2.1 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. |

#### 5.84.3 Member Function Documentation

##### 5.84.3.1 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.

##### 5.84.3.2 void TypeExporter::NotifyListeners ( const TypeExporterEvent & *ev* ) [protected]

Notifies progress to all registered listeners.

**Parameters**

| | |
|---|---|
| *ev* | [in] Progress event to be notified. |

##### 5.84.3.3 void TypeExporter::Register ( TypeExporterListener & *tel* )

Registers a new listener.

**Parameters**

| | |
|---|---|
| *tel* | [in] TypeExporterListener to be registered. |

##### 5.84.3.4 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.

##### 5.84.3.5 void TypeExporter::RunProcess ( ) throw sparksee::gdb::IOException, sparksee::gdb::Error) [protected]

Runs export process.

**Exceptions**

| | |
|---|---|
| *IOException* | If bad things happen writting to the RowWriter. |

#### 5.84.3.6 void TypeExporter::SetAttributes ( sparksee::gdb::AttributeList & *attrs* )

Sets the list of Attributes.

**Parameters**

| | |
|---|---|
| *attrs* | [in] Attribute identifiers to be exported |

#### 5.84.3.7 void TypeExporter::SetFrequency ( sparksee::gdb::int32_t *freq* )

Sets the frequency of listener notification.

**Parameters**

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

#### 5.84.3.8 void TypeExporter::SetGraph ( sparksee::gdb::Graph & *graph* )

Sets the graph that will be exported.

**Parameters**

| | |
|---|---|
| *graph* | [in] Graph. |

#### 5.84.3.9 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 |

Referenced by EdgeTypeExporter::EdgeTypeExporter(), and EdgeTypeExporter::SetHeadAttribute().

#### 5.84.3.10 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. |

#### 5.84.3.11 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 |

Referenced by EdgeTypeExporter::EdgeTypeExporter(), and EdgeTypeExporter::SetHeadPosition().

#### 5.84.3.12 void TypeExporter::SetRowWriter ( RowWriter & *rw* )

Sets the output data destination.

**Parameters**

| | |
|---|---|
| *rw* | [in] Input RowWriter. |

#### 5.84.3.13 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 |

Referenced by EdgeTypeExporter::EdgeTypeExporter(), and EdgeTypeExporter::SetTailAttribute().

#### 5.84.3.14 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 |

Referenced by EdgeTypeExporter::EdgeTypeExporter(), and EdgeTypeExporter::SetTailPosition().

#### 5.84.3.15 void TypeExporter::SetType ( sparksee::gdb::type_t *type* )

Sets the type to be exported.

#### Parameters

| | |
|---|---|
| *type* | [in] Type identifier. |

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

- TypeExporter.h

## 5.85 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**

### 5.85.1 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

### 5.85.2 Member Function Documentation

#### 5.85.2.1 sparksee::gdb::int64_t TypeExporterEvent::GetCount ( ) const [inline]

Gets the current number of objects exported.

#### Returns

The current number of objects exported.

#### 5.85.2.2 sparksee::gdb::int64_t TypeExporterEvent::GetTotal ( ) const [inline]

Gets the total number of objects exported.

**Returns**

The total number of objects exported.

#### 5.85.2.3 sparksee::gdb::type_t TypeExporterEvent::GetTypeId ( ) const [inline]

Gets the type identifier.

**Returns**

The type identifier.

#### 5.85.2.4 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.

References operator<<().

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

- TypeExporter.h

## 5.86 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.*

#### 5.86.1 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

#### 5.86.2 Constructor & Destructor Documentation

##### 5.86.2.1 TypeExporterListener::TypeExporterListener ( ) [inline],[protected]

Protected because none should instantiate a RowWriter.

Just inherited classes may use this empty constructor.

#### 5.86.3 Member Function Documentation

##### 5.86.3.1 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

## 5.87 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.*

#### 5.87.1 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

#### 5.87.2 Constructor & Destructor Documentation

##### 5.87.2.1 TypeList::TypeList ( )

Constructor.

This creates an empty list.

##### 5.87.2.2 TypeList::TypeList ( const std::vector< type_t > & *v* )

Constructor.

**Parameters**

| | |
|---|---|
| *v* | [in] Vector. |

#### 5.87.3 Member Function Documentation

##### 5.87.3.1 void TypeList::Add ( type_t *type* ) [inline]

Adds a Sparksee type identifier at the end of the list.

**Parameters**

| | |
|---|---|
| *type* | [in] Sparksee type identifier. |

#### 5.87.3.2 int32_t TypeList::Count ( ) const [inline]

Number of elements in the list.

**Returns**

Number of elements in the list.

#### 5.87.3.3 TypeListIterator* TypeList::Iterator ( )

Gets a new TypeListIterator.

**Returns**

TypeListIterator instance.

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

- Graph_data.h

## 5.88 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**

### 5.88.1 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

#### 5.88.2 Member Function Documentation

##### 5.88.2.1 bool_t TypeListIterator::HasNext ( ) [inline]

Gets if there are more elements.

**Returns**

TRUE if there are more elements, FALSE otherwise.

##### 5.88.2.2 type_t TypeListIterator::Next ( ) [inline]

Moves to the next element.

**Returns**

The next element.

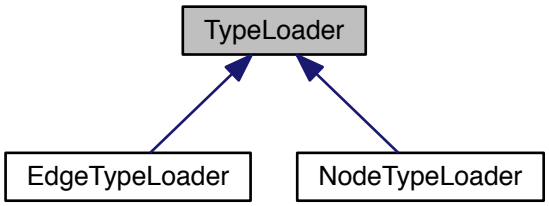
The documentation for this class was generated from the following file:

- Graph_data.h

### 5.89 TypeLoader Class Reference

Base TypeLoader class.

Inheritance 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 &graph)

  *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 type)

  *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 &timestampFormat)

  *Sets a specific timestamp format.*
- void SetFrequency (sparksee::gdb::int32_t freq)

  *Sets the frequency of listener notification.*

### Protected Types

### 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.*
- void SetTailMEP (sparksee::gdb::MissingEndpoint mep)

  *Sets the flag to create missing tail nodes when loading nodes or edges.*
- void SetHeadMEP (sparksee::gdb::MissingEndpoint mep)

  *Sets the flag to create missing head nodes when loading nodes or edges.*

#### 5.89.1 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

#### 5.89.2 Member Enumeration Documentation

##### 5.89.2.1 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.

#### 5.89.3 Constructor & Destructor Documentation

##### 5.89.3.1 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) |

### 5.89.4 Member Function Documentation

#### 5.89.4.1 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.

#### 5.89.4.2 void TypeLoader::NotifyListeners ( const TypeLoaderEvent & *ev* ) [protected]

Notifies progress to all registered listeners.

**Parameters**

| | |
|---|---|
| *ev* | [in] Progress event to be notified. |

#### 5.89.4.3 void TypeLoader::Register ( TypeLoaderListener & *tel* )

Registers a new listener.

**Parameters**

| | | |
|---|---|---|
| in | *tel* | TypeLoaderListener to be registered. |

#### 5.89.4.4 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. |

**Parameters**

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

#### 5.89.4.5 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.

#### 5.89.4.6 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.

#### 5.89.4.7 void TypeLoader::SetAttributePositions ( sparksee::gdb::Int32List & *attrsPos* )

Sets the list of attribute positions.

**Parameters**

| | |
|---|---|
| *attrsPos* | [in] Attribute positions (column index >=0). |

#### 5.89.4.8 void TypeLoader::SetAttributes ( sparksee::gdb::AttributeList & *attrs* )

Sets the list of Attributes.

**Parameters**

| | |
|---|---|
| *attrs* | [in] Attribute identifiers to be loaded |

#### 5.89.4.9 void TypeLoader::SetFrequency ( sparksee::gdb::int32_t *freq* )

Sets the frequency of listener notification.

**Parameters**

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

#### 5.89.4.10 void TypeLoader::SetGraph ( sparksee::gdb::Graph & *graph* )

Sets the graph where the data will be loaded.

**Parameters**

| | |
|---|---|
| *graph* | [in] Graph. |

#### 5.89.4.11 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 |

Referenced by EdgeTypeLoader::EdgeTypeLoader(), and EdgeTypeLoader::SetHeadAttribute().

#### 5.89.4.12 void TypeLoader::SetHeadMEP ( sparksee::gdb::MissingEndpoint *mep* ) [protected]

Sets the flag to create missing head nodes when loading nodes or edges.

**Parameters**

| | |
|---|---|
| *flag* | True if the nodes need to be created. False otherwise |

Referenced by EdgeTypeLoader::EdgeTypeLoader(), and EdgeTypeLoader::SetHeadMEP().

#### 5.89.4.13 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 |

Referenced by EdgeTypeLoader::EdgeTypeLoader(), and EdgeTypeLoader::SetHeadPosition().

#### 5.89.4.14 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. |

#### 5.89.4.15 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.

**Parameters**

| | |
|---|---|
| *path* | [in] The path to the error log file. |

**Exceptions**

| | |
|---|---|
| *IOException* | If bad things happen opening the file. |

#### 5.89.4.16 void TypeLoader::SetLogOff ( )

Truns 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.

#### 5.89.4.17 void TypeLoader::SetRowReader ( RowReader & *rr* )

Sets the input data source.

**Parameters**

| | |
|---|---|
| *rr* | [in] Input RowReader. |

#### 5.89.4.18 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 |

Referenced by EdgeTypeLoader::EdgeTypeLoader(), and EdgeTypeLoader::SetTailAttribute().

#### 5.89.4.19 void TypeLoader::SetTailMEP ( sparksee::gdb::MissingEndpoint *mep* ) [protected]

Sets the flag to create missing tail nodes when loading nodes or edges.

**Parameters**

| | |
|---|---|
| *flag* | True if the nodes need to be created. False otherwise |

Referenced by EdgeTypeLoader::EdgeTypeLoader(), and EdgeTypeLoader::SetTailMEP().

#### 5.89.4.20 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 |

Referenced by EdgeTypeLoader::EdgeTypeLoader(), and EdgeTypeLoader::SetTailPosition().

#### 5.89.4.21 void TypeLoader::SetTimestampFormat ( const std::wstring & *timestampFormat* )

Sets a specific timestamp format.

**Parameters**

| | |
|---|---|
| *timestampFormat* | [in] A string with the timestamp format definition. |

#### 5.89.4.22 void TypeLoader::SetType ( sparksee::gdb::type_t *type* )

Sets the type to be loaded.

**Parameters**

| | |
|---|---|
| *type* | [in] Type identifier. |

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

- TypeLoader.h

## 5.90 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**

### 5.90.1 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

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

- TypeLoader.h

## 5.91 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.*

### 5.91.1 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

### 5.91.2 Constructor & Destructor Documentation

#### 5.91.2.1 TypeLoaderListener::TypeLoaderListener ( ) [inline],[protected]

Protected because none should instantiate a RowWriter.

Just inherited classes may use this empty constructor.

### 5.91.3 Member Function Documentation

#### 5.91.3.1 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. |

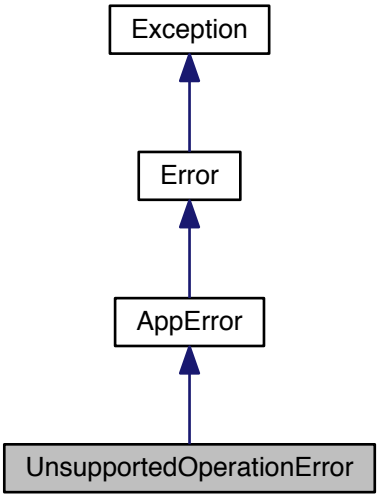
The documentation for this class was generated from the following file:

- TypeLoader.h

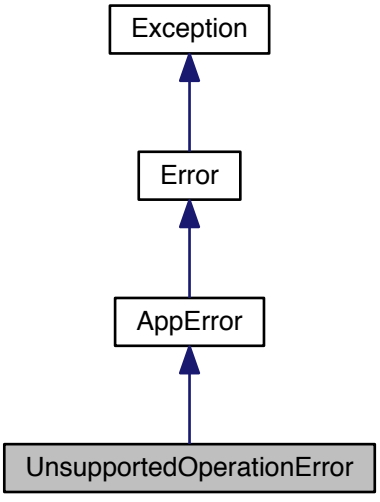
## 5.92 UnsupportedOperationError Class Reference

Unsupported operation error class.

Inheritance diagram for UnsupportedOperationError:




Collaboration diagram for UnsupportedOperationError:




**Public Member Functions**

- UnsupportedOperationError ()

  *Creates a new instance.*

- UnsupportedOperationError (const std::string &mess)

  *Creates a new instance.*
- virtual ~UnsupportedOperationError ()

  *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 void ThrowError (int32_t coreErrorCode)

  *Throws a new Error instance from a sparksee_core error code.*

#### Protected Attributes

- std::string message

  *Message of the exception.*

### 5.92.1 Detailed Description

Unsupported operation error class.

#### Author

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

### 5.92.2 Constructor & Destructor Documentation

#### 5.92.2.1 UnsupportedOperationError::UnsupportedOperationError ( const std::string & *mess* )

Creates a new instance.

#### Parameters

| | |
|---|---|
| *mess* | [in] Message of the exception. |

### 5.92.3 Member Function Documentation

#### 5.92.3.1 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.

#### 5.92.3.2 void Exception::SetMessage ( const std::string & *mess* ) [inherited]

Sets the message of the exception.

**Parameters**

| | |
|---|---|
| *mess* | [in] Message. |

#### 5.92.3.3 static void Error::ThrowError ( int32_t *coreErrorCode* ) [static], [inherited]

Throws 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 will throw an Error instance of an specific Error subclass instance.

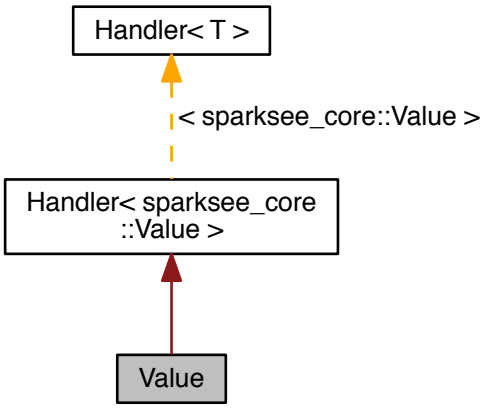
The documentation for this class was generated from the following file:

- Exception.h

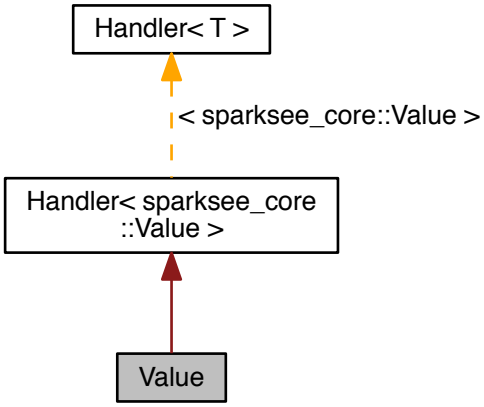
## 5.93 Value Class Reference

Value class.

Inheritance diagram for Value:



Collaboration diagram for Value:




**Public Member Functions**

- Value ()
  *Creates a new instance.*
- Value (const Value &value)
  *Copy constructor.*
- virtual ~Value ()
  *Destructor.*
- Value & operator= (const Value &value)
  *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 value)

  *Sets the Value.*
- Value & SetBoolean (bool_t value)

  *Sets the Value.*
- void SetIntegerVoid (int32_t value)

  *Sets the Value.*
- Value & SetInteger (int32_t value)

  *Sets the Value.*
- void SetLongVoid (int64_t value)

  *Sets the Value.*
- Value & SetLong (int64_t value)

  *Sets the Value.*
- void SetDoubleVoid (double64_t value)

  *Sets the Value.*
- Value & SetDouble (double64_t value)

  *Sets the Value.*
- void SetTimestampVoid (int64_t value)

  *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 value)

  *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 &value)

  *Sets the Value.*
- Value & SetString (const std::wstring &value)

  *Sets the Value.*
- void SetOIDVoid (oid_t value)

  *Sets the OID Value.*
- Value & SetOID (oid_t value)

  *Sets the Value.*
- void SetVoid (Value &value)

  *Sets the Value.*
- Value & Set (Value &value)

  *Sets the Value.*
- int32_t Compare (const Value &value) const

  *Compares with the given Value.*
- bool_t Equals (const Value &value) 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.*

#### Private Member Functions

- sparksee_core::Value * GetHandler ()

  *Gets the handled reference.*
- const sparksee_core::Value * GetHandler () const

  *Gets the handled reference.*
- void SetHandler (sparksee_core::Value *h)

  *Sets the handled reference.*
- void FreeHandler ()

  *Frees (deletes) the handled reference.*

#### Friends

- class **Graph**
- class **ValuesIterator**
- class **ResultSet**
- class **Query**
- class **KeyValues**
- class **KeyValue**
- class **ValueArray**

### 5.93.1 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

### 5.93.2 Constructor & Destructor Documentation

#### 5.93.2.1 Value::Value ( )

Creates a new instance.

It creates a NULL Value.

#### 5.93.2.2 Value::Value ( const Value & *value* )

Copy constructor.

#### Parameters

| | |
|---|---|
| *value* | [in] Value to be copied. |

#### 5.93.3 Member Function Documentation

##### 5.93.3.1 int32_t Value::Compare ( const Value & *value* ) const

Compares with the given Value.

It does not work if the given Value objects does not have the same DataType.

**Parameters**

| | |
|---|---|
| *value* | 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.

##### 5.93.3.2 bool_t Value::Equals ( const Value & *value* ) const

Compares with the given Value.

It does not work if the given Value objects does not have the same DataType.

**Parameters**

| | |
|---|---|
| *value* | Given value to compare to. |

**Returns**

TRUE if this Value is equal to the given one; FALSE otherwise.

##### 5.93.3.3 bool_t Value::GetBoolean ( ) const

Gets Boolean Value.

This must be a non-NULL Boolean Value.

**Returns**

The Boolean Value.

##### 5.93.3.4 DataType Value::GetDataType ( ) const

Gets the DataType.

Value cannot be NULL.

**Returns**

The DataType.

#### 5.93.3.5 double64_t Value::GetDouble ( ) const

Gets Double Value.

This must be a non-NULL Double Value.

**Returns**

The Double Value.

#### 5.93.3.6 int32_t Value::GetInteger ( ) const

Gets Integer Value.

This must be a non-NULL Integer Value.

**Returns**

The Integer Value.

#### 5.93.3.7 int64_t Value::GetLong ( ) const

Gets Long Value.

This must be a non-NULL Long Value.

**Returns**

The Long Value.

#### 5.93.3.8 oid_t Value::GetOID ( ) const

Gets OID Value.

This must be an non-NULL OID Value.

**Returns**

The OID Value.

#### 5.93.3.9 const std::wstring& Value::GetString ( ) const

Gets String Value.

This must be a non-NULL String Value.

**Returns**

The String Value.

#### 5.93.3.10 int64_t Value::GetTimestamp ( ) const

Gets Timestamp Value.

This must be a non-NULL Timestamp Value.

**Returns**

The Timestamp Value.

#### 5.93.3.11 bool_t Value::IsNull ( ) const

Gets if this is a NULL Value.

**Returns**

TRUE if this is a NULL Value, FALSE otherwise.

#### 5.93.3.12 Value& Value::operator= ( const Value & *value* )

Assignment operator.

**Parameters**

| | |
|---|---|
| *value* | [in] Value to be copied. |

**Returns**

Returns the Value reference.

#### 5.93.3.13 Value& Value::Set ( Value & *value* ) [inline]

Sets the Value.

**Parameters**

| | |
|---|---|
| *value* | [in] New value. |

**Returns**

The calling instance.

#### 5.93.3.14 Value& Value::SetBoolean ( bool_t *value* ) [inline]

Sets the Value.

**Parameters**

| | |
|---|---|
| *value* | [in] Nex Boolean value. |

**Returns**

The calling instance.

#### 5.93.3.15 void Value::SetBooleanVoid ( bool_t *value* )

Sets the Value.

**Parameters**

| | |
|---|---|
| *value* | [in] New Boolean value. |

#### 5.93.3.16 Value& Value::SetDouble ( double64_t *value* ) [inline]

Sets the Value.

**Parameters**

| | |
|---|---|
| *value* | [in] New Double value. |

**Returns**

The calling instance.

#### 5.93.3.17 void Value::SetDoubleVoid ( double64_t *value* )

Sets the Value.

**Parameters**

| | |
|---|---|
| *value* | [in] New Double value. |

#### 5.93.3.18 Value& Value::SetInteger ( int32_t *value* ) [inline]

Sets the Value.

**Parameters**

| | |
|---|---|
| *value* | [in] New Integer value. |

**Returns**

The calling instance.

#### 5.93.3.19 void Value::SetIntegerVoid ( int32_t *value* )

Sets the Value.

**Parameters**

| | |
|---|---|
| *value* | [in] New Integer value. |

#### 5.93.3.20 Value& Value::SetLong ( int64_t *value* ) [inline]

Sets the Value.

**Parameters**

| | |
|---|---|
| *value* | [in] New Long value. |

**Returns**

The calling instance.

#### 5.93.3.21 void Value::SetLongVoid ( int64_t *value* )

Sets the Value.

**Parameters**

| | |
|---|---|
| *value* | [in] New Long value. |

#### 5.93.3.22 Value& Value::SetNull ( ) [inline]

Sets the Value to NULL.

**Returns**

The calling instance.

References ValueArray::GetBoolean(), ValueArray::GetDouble(), ValueArray::GetInteger(), ValueArray::GetLong(), ValueArray::GetOID(), and ValueArray::GetTimestamp().

#### 5.93.3.23 Value& Value::SetOID ( oid_t *value* ) [inline]

Sets the Value.

**Parameters**

| | |
|---|---|
| *value* | [in] New OID Value. |

**Returns**

The calling instance.

#### 5.93.3.24 void Value::SetOIDVoid ( oid_t *value* )

Sets the OID Value.

**Parameters**

| | |
|---|---|
| *value* | [in] New OID value. |

#### 5.93.3.25 Value& Value::SetString ( const std::wstring & *value* ) [inline]

Sets the Value.

**Parameters**

| | |
|---|---|
| *value* | [in] New String value. |

**Returns**

The calling instance.

#### 5.93.3.26 void Value::SetStringVoid ( const std::wstring & *value* )

Sets the Value.

**Parameters**

| | |
|---|---|
| *value* | [in] New String value. |

#### 5.93.3.27 Value& Value::SetTimestamp ( int64_t *value* ) [inline]

Sets the Value.

**Parameters**

| | |
|---|---|
| *value* | [in] New Timestamp value. |

**Returns**

The calling instance.

#### 5.93.3.28 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). |

**Parameters**

| | |
|---|---|
| *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.

#### 5.93.3.29 void Value::SetTimestampVoid ( int64_t *value* )

Sets the Value.

**Parameters**

| | |
|---|---|
| *value* | [in] New Timestamp value. |

#### 5.93.3.30 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]). |

#### 5.93.3.31 void Value::SetVoid ( Value & *value* ) [inline]

Sets the Value.

**Parameters**

| | |
|---|---|
| *value* | [in] New value. |

References Handler< T >::GetHandler().

#### 5.93.3.32 std::wstring& Value::ToString ( std::wstring & *str* ) const

Gets a string representation of the Value.

**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.

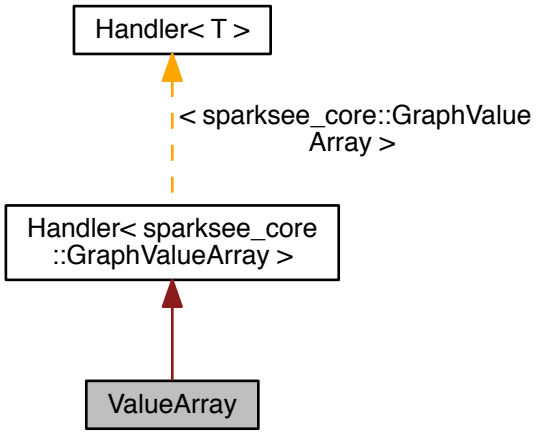
The documentation for this class was generated from the following file:

- Value.h

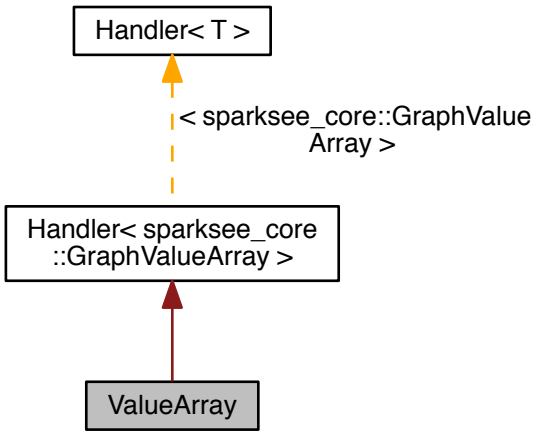
## 5.94 ValueArray Class Reference

ValueArray class.

Inheritance diagram for ValueArray:



Collaboration diagram for ValueArray:




**Public Member Functions**

- void Get (int32_t index, Value &value) const throw (sparksee::gdb::WrongArgumentError, sparksee::gdb::↩ AppError)

  *Get a Value from the array.*
- void Set (int32_t index, const Value &value) throw (sparksee::gdb::WrongArgumentError, sparksee::gdb::↩ AppError)

  *Set a Value to a specific array position.*
- void Set (const Value &value) throw (sparksee::gdb::WrongArgumentError, sparksee::gdb::AppError)

  *Set a Value to the whole array.*
- void MakeNull () throw (sparksee::gdb::AppError)

  *Sets the attribute array to Null.*
- virtual ~ValueArray ()

  *Destructor.*
- int32_t Size () const

  *Returns the array size.*
- void GetDouble (std::vector< sparksee::gdb::double64_t > &values) const throw (sparksee::gdb::AppError, sparksee::gdb::UnsupportedOperationError)

  *Get all the values from this double array.*
- void GetDouble (int32_t index, int32_t size, std::vector< sparksee::gdb::double64_t > &values) const throw (sparksee::gdb::WrongArgumentError, sparksee::gdb::UnsupportedOperationError, sparksee::gdb::↩ AppError)

  *Get a subset of the values from this double array.*
- void SetDouble (const std::vector< sparksee::gdb::double64_t > &values) throw (sparksee::gdb::↩ UnsupportedOperationError, sparksee::gdb::AppError)

  *Set all the values of this double array.*
- void SetDouble (int32_t index, const std::vector< sparksee::gdb::double64_t > &values) throw (sparksee↩ ::gdb::WrongArgumentError, sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

  *Set a subset of the values of this double array.*

- void GetInteger (std::vector< sparksee::gdb::int32_t > &values) const throw (sparksee::gdb::AppError, sparksee::gdb::UnsupportedOperationError)

  *Get all the values from this int array.*
- void GetInteger (int32_t index, int32_t size, std::vector< sparksee::gdb::int32_t > &values) const throw (sparksee::gdb::WrongArgumentError, sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

  *Get a subset of the values from this int array.*
- void SetInteger (const std::vector< sparksee::gdb::int32_t > &values) throw (sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

  *Set all the values of this int array.*
- void SetInteger (int32_t index, const std::vector< sparksee::gdb::int32_t > &values) throw (sparksee::gdb::WrongArgumentError, sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

  *Set a subset of the values of this int array.*
- void GetLong (std::vector< sparksee::gdb::int64_t > &values) const throw (sparksee::gdb::AppError, sparksee::gdb::UnsupportedOperationError)

  *Get all the values from this long array.*
- void GetLong (int32_t index, int32_t size, std::vector< sparksee::gdb::int64_t > &values) const throw (sparksee::gdb::WrongArgumentError, sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

  *Get a subset of the values from this long array.*
- void SetLong (const std::vector< sparksee::gdb::int64_t > &values) throw (sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

  *Set all the values of this long array.*
- void SetLong (int32_t index, const std::vector< sparksee::gdb::int64_t > &values) throw (sparksee::gdb::WrongArgumentError, sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

  *Set a subset of the values of this long array.*
- void GetBoolean (std::vector< sparksee::gdb::bool_t > &values) const throw (sparksee::gdb::AppError, sparksee::gdb::UnsupportedOperationError)

  *Get all the values from this bool array.*
- void GetBoolean (int32_t index, int32_t size, std::vector< sparksee::gdb::bool_t > &values) const throw (sparksee::gdb::WrongArgumentError, sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

  *Get a subset of the values from this bool array.*
- void SetBoolean (const std::vector< sparksee::gdb::bool_t > &values) throw (sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

  *Set all the values of this bool array.*
- void SetBoolean (int32_t index, const std::vector< sparksee::gdb::bool_t > &values) throw (sparksee::gdb::WrongArgumentError, sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

  *Set a subset of the values of this bool array.*
- void GetTimestamp (std::vector< sparksee::gdb::int64_t > &values) const throw (sparksee::gdb::AppError, sparksee::gdb::UnsupportedOperationError)

  *Get all the values from this timestamp array.*
- void GetTimestamp (int32_t index, int32_t size, std::vector< sparksee::gdb::int64_t > &values) const throw (sparksee::gdb::WrongArgumentError, sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

  *Get a subset of the values from this timestamp array.*
- void SetTimestamp (const std::vector< sparksee::gdb::int64_t > &values) throw (sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

  *Set all the values of this timestamp array.*
- void SetTimestamp (int32_t index, const std::vector< sparksee::gdb::int64_t > &values) throw (sparksee::gdb::WrongArgumentError, sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

  *Set a subset of the values of this timestamp array.*
- void GetOID (std::vector< sparksee::gdb::oid_t > &values) const throw (sparksee::gdb::AppError, sparksee::gdb::UnsupportedOperationError)

*Get all the values from this oid array.*

- void GetOID (int32_t index, int32_t size, std::vector< sparksee::gdb::oid_t > &values) const throw (sparksee::gdb::WrongArgumentError, sparksee::gdb::UnsupportedOperationError, sparksee::gdb::App↩ Error)

  *Get a subset of the values from this oid array.*
- void SetOID (const std::vector< sparksee::gdb::oid_t > &values) throw (sparksee::gdb::Unsupported↩ OperationError, sparksee::gdb::AppError)

  *Set all the values of this oid array.*
- void SetOID (int32_t index, const std::vector< sparksee::gdb::oid_t > &values) throw (sparksee::gdb::↩ WrongArgumentError, sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

  *Set a subset of the values of this oid array.*

#### Private Member Functions

- sparksee_core::GraphValueArray * GetHandler ()

  *Gets the handled reference.*
- const sparksee_core::GraphValueArray * GetHandler () const

  *Gets the handled reference.*
- void SetHandler (sparksee_core::GraphValueArray *h)

  *Sets the handled reference.*
- void FreeHandler ()

  *Frees (deletes) the handled reference.*

#### Friends

- class **Graph**

### 5.94.1 Detailed Description

ValueArray class.

It allows for getting and setting ValueArray attribute values.

It is very important to close the ValueArray once no more get or set operations will be performed.

Creation of a new ValueArray: (i) Set all the ValueArray elements using Graph::SetAttributeArray (ii) perform as many get/set operations as you need to the ValueArray instance. Lastly, (iii) Close the ValueArray

Use of an existing ValueArray: (i) Get a ValueArray instance using Graph::GetAttributeArray (ii) perform as many get/set operations as you need to the ValueArray instance. Lastly, (iii) Close the ValueArray

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

### 5.94.2 Member Function Documentation

#### 5.94.2.1 void ValueArray::Get ( int32_t *index*, Value & *value* ) const throw sparksee::gdb::WrongArgumentError, sparksee::gdb::AppError)

Get a Value from the array.

**Parameters**

| | |
|---|---|
| *index* | [in] Position of the element to get [0..N-1] |
| *value* | [out] Value to get the array element |

**Exceptions**

| | |
|---|---|
| *WrongArgumentError* | If the index is out of range or the Value not valid |
| *AppError* | If the ValueArray is not available |

#### 5.94.2.2 void ValueArray::GetBoolean ( std::vector< sparksee::gdb::bool_t > & *values* ) const throw sparksee::gdb::AppError, sparksee::gdb::UnsupportedOperationError)

Get all the values from this bool array.

**Parameters**

| | |
|---|---|
| *values* | [out] All the values will be returned here |

**Exceptions**

| | |
|---|---|
| *AppError* | If the ValueArray is not available |
| *UnsupportedOperationError* | If array DataType is not bool |

Referenced by Value::SetNull().

#### 5.94.2.3 void ValueArray::GetBoolean ( int32_t *index*, int32_t *size*, std::vector< sparksee::gdb::bool_t > & *values* ) const throw sparksee::gdb::WrongArgumentError, sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

Get a subset of the values from this bool array.

**Parameters**

| | |
|---|---|
| *index* | [in] Position of the first element to get [0..N-1] |
| *size* | [in] Number of elements to get [1..N] |
| *values* | [out] All the requested values will be returned here |

**Exceptions**

| | |
|---|---|
| *WrongArgumentError* | If the index or size is out of range |
| *UnsupportedOperationError* | If array DataType is not bool |
| *AppError* | If the ValueArray is not available |

#### 5.94.2.4 void ValueArray::GetDouble ( std::vector< sparksee::gdb::double64_t > & *values* ) const throw sparksee::gdb::AppError, sparksee::gdb::UnsupportedOperationError)

Get all the values from this double array.

**Parameters**

| | |
|---|---|
| *values* | [out] All the values will be returned here |

**Exceptions**

| | |
|---|---|
| *AppError* | If the ValueArray is not available |
| *UnsupportedOperationError* | If array DataType is not Double |

Referenced by Value::SetNull().

#### 5.94.2.5 void ValueArray::GetDouble ( int32_t *index*, int32_t *size*, std::vector< sparksee::gdb::double64_t > & *values* ) const throw sparksee::gdb::WrongArgumentError, sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

Get a subset of the values from this double array.

**Parameters**

| | |
|---|---|
| *index* | [in] Position of the first element to get [0..N-1] |
| *size* | [in] Number of elements to get [1..N] |
| *values* | [out] All the requested values will be returned here |

**Exceptions**

| | |
|---|---|
| *WrongArgumentError* | If the index or size is out of range |
| *UnsupportedOperationError* | If array DataType is not Double |
| *AppError* | If the ValueArray is not available |

#### 5.94.2.6 void ValueArray::GetInteger ( std::vector< sparksee::gdb::int32_t > & *values* ) const throw sparksee::gdb::AppError, sparksee::gdb::UnsupportedOperationError)

Get all the values from this int array.

**Parameters**

| | |
|---|---|
| *values* | [out] All the values will be returned here |

**Exceptions**

| | |
|---|---|
| *AppError* | If the ValueArray is not available |
| *UnsupportedOperationError* | If array DataType is not int |

Referenced by Value::SetNull().

#### 5.94.2.7 void ValueArray::GetInteger ( int32_t *index*, int32_t *size*, std::vector< sparksee::gdb::int32_t > & *values* ) const throw sparksee::gdb::WrongArgumentError, sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

Get a subset of the values from this int array.

**Parameters**

| | |
|---|---|
| *index* | [in] Position of the first element to get [0..N-1] |
| *size* | [in] Number of elements to get [1..N] |
| *values* | [out] All the requested values will be returned here |

**Exceptions**

| | |
|---|---|
| *WrongArgumentError* | If the index or size is out of range |
| *UnsupportedOperationError* | If array DataType is not int |
| *AppError* | If the ValueArray is not available |

#### 5.94.2.8 void ValueArray::GetLong ( std::vector< sparksee::gdb::int64_t > & *values* ) const throw sparksee::gdb::AppError, sparksee::gdb::UnsupportedOperationError)

Get all the values from this long array.

**Parameters**

| | |
|---|---|
| *values* | [out] All the values will be returned here |

**Exceptions**

| | |
|---|---|
| *AppError* | If the ValueArray is not available |
| *UnsupportedOperationError* | If array DataType is not long |

Referenced by Value::SetNull().

#### 5.94.2.9 void ValueArray::GetLong ( int32_t *index*, int32_t *size*, std::vector< sparksee::gdb::int64_t > & *values* ) const throw sparksee::gdb::WrongArgumentError, sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

Get a subset of the values from this long array.

**Parameters**

| | |
|---|---|
| *index* | [in] Position of the first element to get [0..N-1] |
| *size* | [in] Number of elements to get [1..N] |
| *values* | [out] All the requested values will be returned here |

**Exceptions**

| | |
|---|---|
| *WrongArgumentError* | If the index or size is out of range |

**Exceptions**

| | |
|---|---|
| *UnsupportedOperationError* | If array DataType is not long |
| *AppError* | If the ValueArray is not available |

#### 5.94.2.10 void ValueArray::GetOID ( std::vector< sparksee::gdb::oid_t > & *values* ) const throw sparksee::gdb::AppError, sparksee::gdb::UnsupportedOperationError)

Get all the values from this oid array.

**Parameters**

| | |
|---|---|
| *values* | [out] All the values will be returned here |

**Exceptions**

| | |
|---|---|
| *AppError* | If the ValueArray is not available |
| *UnsupportedOperationError* | If array DataType is not oid |

Referenced by Value::SetNull().

#### 5.94.2.11 void ValueArray::GetOID ( int32_t *index*, int32_t *size*, std::vector< sparksee::gdb::oid_t > & *values* ) const throw sparksee::gdb::WrongArgumentError, sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

Get a subset of the values from this oid array.

**Parameters**

| | |
|---|---|
| *index* | [in] Position of the first element to get [0..N-1] |
| *size* | [in] Number of elements to get [1..N] |
| *values* | [out] All the requested values will be returned here |

**Exceptions**

| | |
|---|---|
| *WrongArgumentError* | If the index or size is out of range |
| *UnsupportedOperationError* | If array DataType is not oid |
| *AppError* | If the ValueArray is not available |

#### 5.94.2.12 void ValueArray::GetTimestamp ( std::vector< sparksee::gdb::int64_t > & *values* ) const throw sparksee::gdb::AppError, sparksee::gdb::UnsupportedOperationError)

Get all the values from this timestamp array.

**Parameters**

| | |
|---|---|
| *values* | [out] All the values will be returned here |

**Exceptions**

| | |
|---|---|
| *AppError* | If the ValueArray is not available |
| *UnsupportedOperationError* | If array DataType is not timestamp |

Referenced by Value::SetNull().

#### 5.94.2.13 void ValueArray::GetTimestamp ( int32_t *index*, int32_t *size*, std::vector< sparksee::gdb::int64_t > & *values* ) const throw sparksee::gdb::WrongArgumentError, sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

Get a subset of the values from this timestamp array.

**Parameters**

| | |
|---|---|
| *index* | [in] Position of the first element to get [0..N-1] |
| *size* | [in] Number of elements to get [1..N] |
| *values* | [out] All the requested values will be returned here |

**Exceptions**

| | |
|---|---|
| *WrongArgumentError* | If the index or size is out of range |
| *UnsupportedOperationError* | If array DataType is not timestamp |
| *AppError* | If the ValueArray is not available |

#### 5.94.2.14 void ValueArray::MakeNull ( ) throw sparksee::gdb::AppError)

Sets the attribute array to Null.

The ValueArray can not be used after this call.

#### 5.94.2.15 void ValueArray::Set ( int32_t *index*, const Value & *value* ) throw sparksee::gdb::WrongArgumentError, sparksee::gdb::AppError)

Set a Value to a specific array position.

**Parameters**

| | |
|---|---|
| *index* | [in] Position of the element to set [0..N-1] |
| *value* | [in] Value to set in the array element |

**Exceptions**

| | |
|---|---|
| *WrongArgumentError* | If the index is out of range or the Value not valid |
| *AppError* | If the ValueArray is not available |

#### 5.94.2.16 void ValueArray::Set ( const Value & *value* ) throw sparksee::gdb::WrongArgumentError, sparksee::gdb::AppError)

Set a Value to the whole array.

**Parameters**

| | |
|---|---|
| *value* | [in] Value to set in all the array elements |

**Exceptions**

| | |
|---|---|
| *WrongArgumentError* | If the Value is not valid |
| *AppError* | If the ValueArray is not available |

#### 5.94.2.17 void ValueArray::SetBoolean ( const std::vector< sparksee::gdb::bool_t > & *values* ) throw sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

Set all the values of this bool array.

**Parameters**

| | |
|---|---|
| *values* | [in] All the values to set |

**Exceptions**

| | |
|---|---|
| *AppError* | If the ValueArray is not available |
| *UnsupportedOperationError* | If array DataType is not bool |

#### 5.94.2.18 void ValueArray::SetBoolean ( int32_t *index*, const std::vector< sparksee::gdb::bool_t > & *values* ) throw sparksee::gdb::WrongArgumentError, sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

Set a subset of the values of this bool array.

**Parameters**

| | |
|---|---|
| *index* | [in] Position of the first element to set [0..N-1] |
| *values* | [in] All the values to set |

**Exceptions**

| | |
|---|---|
| *WrongArgumentError* | If the index or size is out of range |
| *UnsupportedOperationError* | If array DataType is not bool |
| *AppError* | If the ValueArray is not available |

#### 5.94.2.19 void ValueArray::SetDouble ( const std::vector< sparksee::gdb::double64_t > & *values* ) throw sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

Set all the values of this double array.

**Parameters**

| | |
|---|---|
| *values* | [in] All the values to set |

**Exceptions**

| | |
|---|---|
| *AppError* | If the ValueArray is not available |
| *UnsupportedOperationError* | If array DataType is not Double |

### 5.94.2.20 void ValueArray::SetDouble ( int32_t *index*, const std::vector< sparksee::gdb::double64_t > & *values* ) throw sparksee::gdb::WrongArgumentError, sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

Set a subset of the values of this double array.

**Parameters**

| | |
|---|---|
| *index* | [in] Position of the first element to set [0..N-1] |
| *values* | [in] All the values to set |

**Exceptions**

| | |
|---|---|
| *WrongArgumentError* | If the index or size is out of range |
| *UnsupportedOperationError* | If array DataType is not Double |
| *AppError* | If the ValueArray is not available |

### 5.94.2.21 void ValueArray::SetInteger ( const std::vector< sparksee::gdb::int32_t > & *values* ) throw sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

Set all the values of this int array.

**Parameters**

| | |
|---|---|
| *values* | [in] All the values to set |

**Exceptions**

| | |
|---|---|
| *AppError* | If the ValueArray is not available |
| *UnsupportedOperationError* | If array DataType is not int |

### 5.94.2.22 void ValueArray::SetInteger ( int32_t *index*, const std::vector< sparksee::gdb::int32_t > & *values* ) throw sparksee::gdb::WrongArgumentError, sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

Set a subset of the values of this int array.

**Parameters**

| | |
|---|---|
| *index* | [in] Position of the first element to set [0..N-1] |
| *values* | [in] All the values to set |

**Exceptions**

| | |
|---|---|
| *WrongArgumentError* | If the index or size is out of range |
| *UnsupportedOperationError* | If array DataType is not int |
| *AppError* | If the ValueArray is not available |

#### 5.94.2.23 void ValueArray::SetLong ( const std::vector< sparksee::gdb::int64_t > & *values* ) throw sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

Set all the values of this long array.

**Parameters**

| | |
|---|---|
| *values* | [in] All the values to set |

**Exceptions**

| | |
|---|---|
| *AppError* | If the ValueArray is not available |
| *UnsupportedOperationError* | If array DataType is not long |

#### 5.94.2.24 void ValueArray::SetLong ( int32_t *index*, const std::vector< sparksee::gdb::int64_t > & *values* ) throw sparksee::gdb::WrongArgumentError, sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

Set a subset of the values of this long array.

**Parameters**

| | |
|---|---|
| *index* | [in] Position of the first element to set [0..N-1] |
| *values* | [in] All the values to set |

**Exceptions**

| | |
|---|---|
| *WrongArgumentError* | If the index or size is out of range |
| *UnsupportedOperationError* | If array DataType is not long |
| *AppError* | If the ValueArray is not available |

#### 5.94.2.25 void ValueArray::SetOID ( const std::vector< sparksee::gdb::oid_t > & *values* ) throw sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

Set all the values of this oid array.

**Parameters**

| | |
|---|---|
| *values* | [in] All the values to set |

**Exceptions**

| | |
|---|---|
| *AppError* | If the ValueArray is not available |
| *UnsupportedOperationError* | If array DataType is not oid |

#### 5.94.2.26 void ValueArray::SetOID ( int32_t *index*, const std::vector< sparksee::gdb::oid_t > & *values* ) throw sparksee::gdb::WrongArgumentError, sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

Set a subset of the values of this oid array.

**Parameters**

| | |
|---|---|
| *index* | [in] Position of the first element to set [0..N-1] |
| *values* | [in] All the values to set |

**Exceptions**

| | |
|---|---|
| *WrongArgumentError* | If the index or size is out of range |
| *UnsupportedOperationError* | If array DataType is not oid |
| *AppError* | If the ValueArray is not available |

#### 5.94.2.27 void ValueArray::SetTimestamp ( const std::vector< sparksee::gdb::int64_t > & *values* ) throw sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

Set all the values of this timestamp array.

**Parameters**

| | |
|---|---|
| *values* | [in] All the values to set |

**Exceptions**

| | |
|---|---|
| *AppError* | If the ValueArray is not available |
| *UnsupportedOperationError* | If array DataType is not timestamp |

#### 5.94.2.28 void ValueArray::SetTimestamp ( int32_t *index*, const std::vector< sparksee::gdb::int64_t > & *values* ) throw sparksee::gdb::WrongArgumentError, sparksee::gdb::UnsupportedOperationError, sparksee::gdb::AppError)

Set a subset of the values of this timestamp array.

**Parameters**

| | |
|---|---|
| *index* | [in] Position of the first element to set [0..N-1] |
| *values* | [in] All the values to set |

#### Exceptions

| | |
|---|---|
| *WrongArgumentError* | If the index or size is out of range |
| *UnsupportedOperationError* | If array DataType is not timestamp |
| *AppError* | If the ValueArray is not available |

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

- ValueArray.h

## 5.95 ValueList Class Reference

Value list.

#### Public Member Functions

- int32_t Count () const
  *Number of elements in the list.*
- ValueListIterator * Iterator ()
  *Gets a new ValueListIterator.*
- ValueList ()
  *Constructor.*
- ~ValueList ()
  *Destructor.*
- void Clear ()
  *Clears the list.*
- void Add (Value *value)
  *Adds a value to the end of the list.*
- Value * Get (int32_t index) const
  *Returns the Value at the specified position in the list.*

### 5.95.1 Detailed Description

Value list.

It stores a Value list.

Use ValueListIterator to access all elements into this collection.

#### Author

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

### 5.95.2 Constructor & Destructor Documentation

#### 5.95.2.1 ValueList::ValueList ( ) [inline]

Constructor.

This creates an empty list.

### 5.95.3 Member Function Documentation

#### 5.95.3.1 void ValueList::Add ( Value * *value* )

Adds a value to the end of the list.

**Parameters**

| | |
|---|---|
| *value* | [in] The value to add |

#### 5.95.3.2 int32_t ValueList::Count ( ) const [inline]

Number of elements in the list.

**Returns**

Number of elements in the list.

#### 5.95.3.3 Value* ValueList::Get ( int32_t *index* ) const

Returns the Value at the specified position in the list.

**Parameters**

| | |
|---|---|
| *index* | [in] Index of the element to return, starting at 0. |

#### 5.95.3.4 ValueListIterator* ValueList::Iterator ( )

Gets a new ValueListIterator.

**Returns**

ValueListIterator instance.

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

- Value.h

## 5.96 ValueListIterator Class Reference

ValueList iterator class.

**Public Member Functions**

- ~ValueListIterator ()
  *Destructor.*
- const Value * Next ()
  *Moves to the next element.*
- bool_t HasNext ()
  *Gets if there are more elements.*

#### Friends

- class **ValueList**

### 5.96.1 Detailed Description

ValueList iterator class.

Iterator to traverse all the values into a ValueList instance.

#### Author

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

### 5.96.2 Member Function Documentation

#### 5.96.2.1 bool_t ValueListIterator::HasNext ( ) [inline]

Gets if there are more elements.

#### Returns

TRUE if there are more elements, FALSE otherwise.

References END_SPARKSEE_GDB_NAMESPACE.

#### 5.96.2.2 const Value* ValueListIterator::Next ( ) [inline]

Moves to the next element.

#### Returns

The next element.

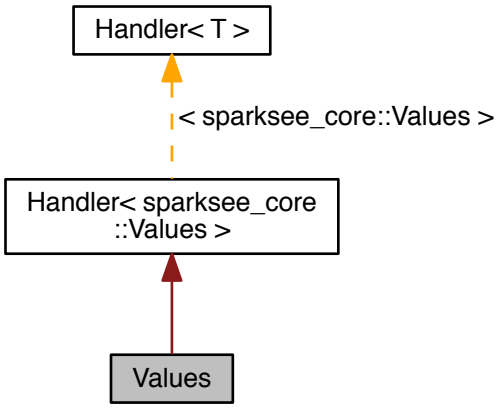
The documentation for this class was generated from the following file:

- Value.h

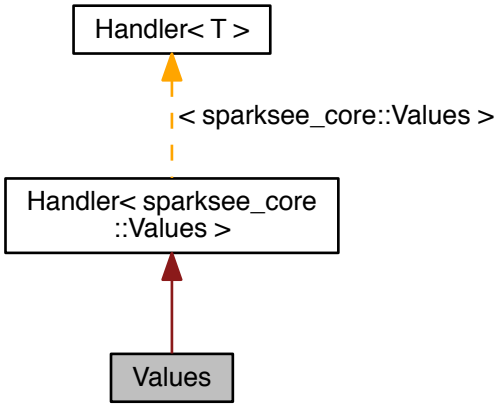
## 5.97 Values Class Reference

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.*

#### Private Member Functions

- sparksee_core::Values * GetHandler ()

  *Gets the handled reference.*
- const sparksee_core::Values * GetHandler () const

  *Gets the handled reference.*
- void SetHandler (sparksee_core::Values *h)

  *Sets the handled reference.*
- void FreeHandler ()

  *Frees (deletes) the handled reference.*
- bool_t IsNull () const

  *Gets if the handler is NULL.*

#### Friends

- class **Graph**
- class **ValuesIterator**

### 5.97.1 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

### 5.97.2 Member Function Documentation

#### 5.97.2.1 int64_t Values::Count ( )

Gets the number of elements into the collection.

#### Returns

The number of elements into the collection.

#### 5.97.2.2 ValuesIterator* Values::Iterator ( Order *order* )

Gets a ValuesIterator.

#### Returns

ValuesIterator instance.

**Parameters**

| | |
|---|---|
| *order* | [in] Ascending or descending order. |

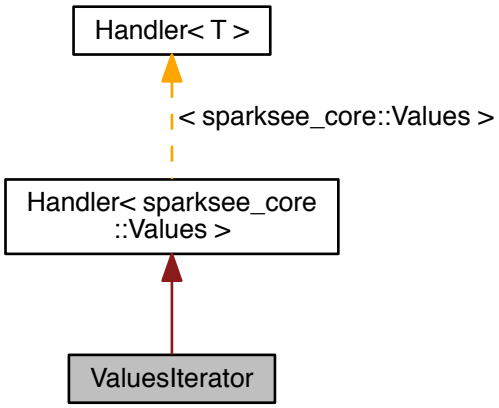
The documentation for this class was generated from the following file:

- Values.h

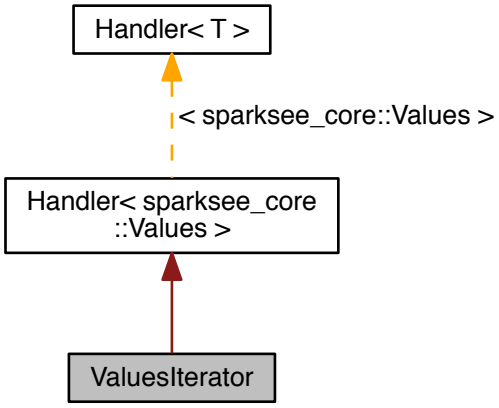
## 5.98 ValuesIterator Class Reference

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.*

#### Private Member Functions

- sparksee_core::Values * GetHandler ()

  *Gets the handled reference.*
- const sparksee_core::Values * GetHandler () const

  *Gets the handled reference.*
- void SetHandler (sparksee_core::Values *h)

  *Sets the handled reference.*
- void FreeHandler ()

  *Frees (deletes) the handled reference.*
- bool_t IsNull () const

  *Gets if the handler is NULL.*

#### Friends

- class **Values**

### 5.98.1 Detailed Description

Values iterator class.

It allows for traversing all the elements into a Values instance.

#### Author

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

### 5.98.2 Member Function Documentation

#### 5.98.2.1 bool_t ValuesIterator::HasNext ( )

Gets if there are more elements to traverse.

#### Returns

TRUE if there are more elements to traverse, FALSE otherwise.

#### 5.98.2.2 Value* ValuesIterator::Next ( )

Gets the next element to traverse.

**Returns**

The next element.

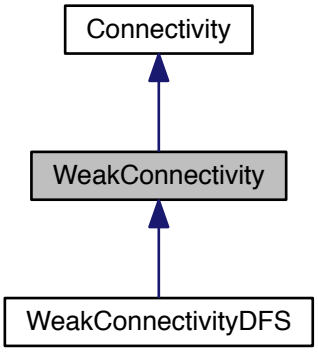
The documentation for this class was generated from the following file:

- ValuesIterator.h

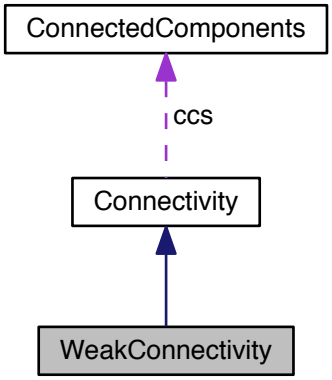
## 5.99 WeakConnectivity Class Reference

WeakConnectivity class.

Inheritance diagram for WeakConnectivity:




Collaboration diagram for WeakConnectivity:



### Public Member Functions

- virtual ~WeakConnectivity ()

  *Destructor.*
- virtual void AddEdgeType (sparksee::gdb::type_t type)

  *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.*

#### 5.99.1 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

#### 5.99.2 Constructor & Destructor Documentation

##### 5.99.2.1 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 |

#### 5.99.3 Member Function Documentation

##### 5.99.3.1 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.

##### 5.99.3.2 void Connectivity::AddAllEdgeTypes ( sparksee::gdb::EdgesDirection *d* ) [protected], [inherited]

Allows connectivity through all edge types of the graph.

**Parameters**

| | |
|---|---|
| *d* | [in] Edge direction. |

#### 5.99.3.3 virtual void WeakConnectivity::AddEdgeType ( sparksee::gdb::type_t *type* ) [virtual]

Allows connectivity through edges of the given type.

In a weak connectivity the edges can be used in Any direction.

**Parameters**

| | |
|---|---|
| *type* | [in] Edge type. |

#### 5.99.3.4 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. |

#### 5.99.3.5 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. |

#### 5.99.3.6 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. |

#### 5.99.3.7 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.

#### 5.99.3.8 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 WeakConnectivityDFS, and StrongConnectivityGabow.

#### 5.99.3.9 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. |

#### 5.99.3.10 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.

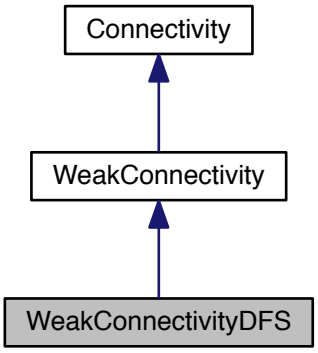
The documentation for this class was generated from the following file:

- WeakConnectivity.h

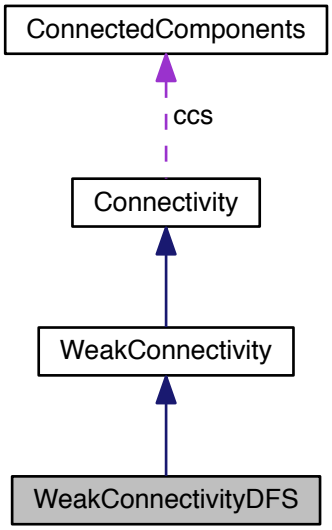
## 5.100 WeakConnectivityDFS Class Reference

WeakConnectivityDFS class.

Inheritance diagram for WeakConnectivityDFS:




Collaboration diagram for WeakConnectivityDFS:




**Public Member Functions**

- WeakConnectivityDFS (sparksee::gdb::Session &session)

  *Creates a new instance of WeakConnectivityDFS.*
- virtual ~WeakConnectivityDFS ()

  *Destructor.*
- void Run ()

*Executes the algorithm.*

- virtual void AddEdgeType (sparksee::gdb::type_t type)

  *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.*

#### 5.100.1 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

#### 5.100.2 Constructor & Destructor Documentation

##### 5.100.2.1 WeakConnectivityDFS::WeakConnectivityDFS ( sparksee::gdb::Session & *session* )

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**

| | |
|---|---|
| *session* | [in] Session to get the graph from and calculate the connectivity |

#### 5.100.3 Member Function Documentation

##### 5.100.3.1 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.

##### 5.100.3.2 void Connectivity::AddAllEdgeTypes ( sparksee::gdb::EdgesDirection *d* ) [protected], [inherited]

Allows connectivity through all edge types of the graph.

**Parameters**

| | |
|---|---|
| *d* | [in] Edge direction. |

#### 5.100.3.3 virtual void WeakConnectivity::AddEdgeType ( sparksee::gdb::type_t *type* ) [virtual], [inherited]

Allows connectivity through edges of the given type.

In a weak connectivity the edges can be used in Any direction.

**Parameters**

| | |
|---|---|
| *type* | [in] Edge type. |

#### 5.100.3.4 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. |

#### 5.100.3.5 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. |

#### 5.100.3.6 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. |

#### 5.100.3.7 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.

#### 5.100.3.8 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. |

#### 5.100.3.9 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.

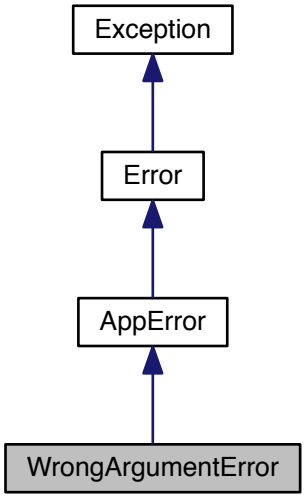
The documentation for this class was generated from the following file:

- WeakConnectivityDFS.h

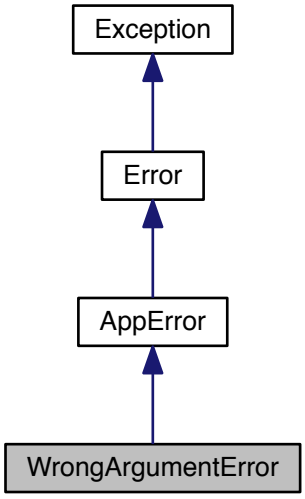
## 5.101 WrongArgumentError Class Reference

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 void ThrowError (int32_t coreErrorCode)

  *Throws a new Error instance from a sparksee_core error code.*

#### Protected Attributes

- std::string message

  *Message of the exception.*

### 5.101.1 Detailed Description

Wrong argument error class.

#### Author

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

### 5.101.2 Constructor & Destructor Documentation

#### 5.101.2.1 WrongArgumentError::WrongArgumentError ( const std::string & *mess* )

Creates a new instance.

#### Parameters

| | |
|---|---|
| *mess* | [in] Message of the exception. |

### 5.101.3 Member Function Documentation

#### 5.101.3.1 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.

#### 5.101.3.2 void Exception::SetMessage ( const std::string & *mess* ) [inherited]

Sets the message of the exception.

**Parameters**

| | |
|---|---|
| *mess* | [in] Message. |

#### 5.101.3.3 static void Error::ThrowError ( int32_t *coreErrorCode* ) [static], [inherited]

Throws 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 will throw an Error instance of an specific Error subclass instance.

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

- Exception.h

# Index


~Handler
Handler, 156

Add
AttributeList, 38
BooleanList, 44
Int32List, 158
OIDList, 201
Objects, 193
StringList, 296
TypeList, 339
ValueList, 377
AddAllEdgeTypes
Algorithms, 29
CommunitiesSCD, 53
CommunityDetection, 58
Connectivity, 66
Context, 69
DisjointCommunityDetection, 95
PageRank, 204
RandomWalk, 220
ShortestPath, 242
SinglePairShortestPath, 247
SinglePairShortestPathBFS, 253
SinglePairShortestPathDijkstra, 258
StrongConnectivity, 301
StrongConnectivityGabow, 306
Traversal, 317
TraversalBFS, 321
TraversalDFS, 326
WeakConnectivity, 387
WeakConnectivityDFS, 393
AddChecksums
Sparksee, 265
AddEdgeType
Algorithms, 30
CommunitiesSCD, 53
CommunityDetection, 58
Connectivity, 66
Context, 69
DisjointCommunityDetection, 95, 96
PageRank, 204
RandomWalk, 220
ShortestPath, 243
SinglePairShortestPath, 247
SinglePairShortestPathBFS, 253
SinglePairShortestPathDijkstra, 259
StrongConnectivity, 301
StrongConnectivityGabow, 306
Traversal, 317
TraversalBFS, 322
TraversalDFS, 326
WeakConnectivity, 387, 388
WeakConnectivityDFS, 394
AddNodeType
Algorithms, 30
RandomWalk, 220
Traversal, 317
TraversalBFS, 322
TraversalDFS, 326
AddWeightedEdgeType
SinglePairShortestPathDijkstra, 259
Algorithms, 27
AddAllEdgeTypes, 29
AddEdgeType, 30
AddNodeType, 30
KOpt, 30
SetCurrentTour, 30
SetEdgeWeightAttributeType, 31
SetMaxIterations, 31
SetTimeLimit, 31
Any
Gdb, 17
Objects, 194
AppError, 32
AppError, 33
Message, 33
SetMessage, 33
ThrowError, 34
AsDirected
EdgeExport, 100
Ascendent
Gdb, 18
Attribute, 34
GetArraySize, 35
GetCount, 35
GetDataType, 35
GetId, 35
GetKind, 36
GetName, 36
GetSize, 36
GetTypeId, 36
IsArrayAttribute, 36
IsSessionAttribute, 36
AttributeKind
Gdb, 15
AttributeList, 37
Add, 38
AttributeList, 37
Count, 38
Iterator, 38
AttributeListIterator, 38
HasNext, 39
Next, 39
AttributeStatistics, 39
GetAvgLengthString, 41
GetDistinct, 41
GetMax, 41
GetMaxLengthString, 41
GetMean, 41

GetMedian, 41
GetMin, 42
GetMinLengthString, 42
GetMode, 42
GetModeCount, 42
GetNull, 42
GetTotal, 43
GetVariance, 43

Backup
Graph, 131
Basic
Gdb, 15
Between
Gdb, 16
Boolean
Gdb, 16
BooleanList, 43
Add, 44
BooleanList, 44
Count, 44
Iterator, 44
BooleanListIterator, 45
HasNext, 45
Next, 45
Box
Gdb, 18
BulkLoader, 46
SetAttributePositions, 47
SetAttributes, 47
SetFilename, 48
SetLocale, 48
SetLogError, 48
SetTimestampFormat, 48
SetType, 48

CSVLoader, 71
LoadEdges, 71
LoadNodes, 72
CSVReader, 73
Close, 74
GetRow, 75
Open, 75
Read, 75
Reset, 76
SetLocale, 76
SetMultilines, 76
SetNumLines, 76
SetQuotes, 76
SetSeparator, 77
SetStartLine, 77
CSVWriter, 77
Close, 79
Open, 79
SetAutoQuotes, 79
SetForcedQuotes, 79
SetLocale, 79
SetQuotes, 80
SetSeparator, 80
Write, 80
CalculateEdgeCost
SinglePairShortestPathDijkstraDynamicCost, 262
CanRun
EdgeTypeExporter, 104
EdgeTypeLoader, 110
NodeTypeExporter, 178
NodeTypeLoader, 183
TypeExporter, 333
TypeLoader, 344
CheckOnlyExistence
SinglePairShortestPathBFS, 253
Close
CSVReader, 74
CSVWriter, 79
RowReader, 231
RowWriter, 234
TextStream, 314
ColorRGB
Gdb, 15
CombineDifference
Objects, 194
CombineIntersection
Objects, 194
CombineUnion
Objects, 195
CommunitiesSCD, 49
AddAllEdgeTypes, 53
AddEdgeType, 53
CommunitiesSCD, 53
ExcludeEdges, 54
ExcludeNodes, 54
GetCommunities, 54
IncludeEdges, 54
IncludeNodes, 55
SetLookAhead, 55
SetMaterializedAttribute, 55
SetNodesNotVisited, 55
CommunityDetection, 56
AddAllEdgeTypes, 58
AddEdgeType, 58
CommunityDetection, 58
ExcludeEdges, 59
ExcludeNodes, 59
IncludeEdges, 59
IncludeNodes, 59
Run, 60
SetNodesNotVisited, 60
Compare
Value, 357
Compute
Context, 69, 70
Condition
Gdb, 15
Config
Gdb, 17
ConnectedComponents, 60
ConnectedComponents, 61

GetConnectedComponent, 61
GetCount, 62
GetNodes, 62
GetSize, 62
Connectivity, 62
AddAllEdgeTypes, 66
AddEdgeType, 66
Connectivity, 65
ExcludeEdges, 66
ExcludeNodes, 66
GetConnectedComponents, 66
Run, 67
SetMaterializedAttribute, 67
SetNodesNotVisited, 67
Contains
Objects, 195
Context, 67
AddAllEdgeTypes, 69
AddEdgeType, 69
Compute, 69, 70
Context, 69
ExcludeEdges, 70
ExcludeNodes, 70
SetMaximumHops, 70
Copy
Objects, 195
Count
AttributeList, 38
BooleanList, 44
Int32List, 158
OIDList, 201
Objects, 196
ResultSetList, 228
StringList, 296
TypeList, 339
ValueList, 378
Values, 381
CountEdges
Graph, 132
CountNodes
Graph, 132
Create
Gdb, 18
Sparksee, 266

DataType
Gdb, 16
Database, 80
GetAlias, 83
GetCacheMaxSize, 83
GetPath, 83
GetStatistics, 83
RedoPrecommitted, 83
SetCacheMaxSize, 83
DatabaseStatistics, 84
GetCache, 84
GetData, 84
GetRead, 85
GetSessions, 85
GetTemp, 85
GetWrite, 85
Debug
Gdb, 17
Decrypt
Sparksee, 267
DefaultExport, 86
EnableType, 87
GetEdge, 87
GetEdgeType, 88
GetGraph, 88
GetNode, 88
GetNodeType, 89
Degree
Graph, 132
Descendent
Gdb, 18
Difference
Objects, 196
DisjointCommunities, 89
DisjointCommunities, 90
GetCommunity, 90
GetCount, 91
GetNodes, 91
GetSize, 91
DisjointCommunityDetection, 91
AddAllEdgeTypes, 95
AddEdgeType, 95, 96
DisjointCommunityDetection, 95
ExcludeEdges, 96
ExcludeNodes, 96
GetCommunities, 96
IncludeEdges, 96
IncludeNodes, 97
Run, 97
SetMaterializedAttribute, 97
SetNodesNotVisited, 98
Double
Gdb, 16
DownloadLicense
SparkseeConfig, 277
Drop
Graph, 132, 133
DumpData
Graph, 133
DumpStorage
Graph, 133

Edge
Gdb, 18
EdgeData, 98
GetEdge, 98
GetHead, 98
GetTail, 99
EdgeExport, 99
AsDirected, 100
GetColorRGB, 100
GetFontSize, 100
GetLabel, 100

GetLabelColorRGB, 101
GetWidth, 101
SetAsDirected, 101
SetColorRGB, 101
SetFontSize, 101
SetLabel, 102
SetLabelColorRGB, 102
SetWidth, 102
EdgeTypeExporter, 102
CanRun, 104
EdgeTypeExporter, 104
NotifyListeners, 105
Register, 105
RunProcess, 105
SetAttributes, 105
SetFrequency, 105
SetGraph, 105
SetHeadAttribute, 106
SetHeadPosition, 106
SetHeader, 106
SetRowWriter, 106
SetTailAttribute, 106
SetTailPosition, 107
SetType, 107
EdgeTypeLoader, 107
CanRun, 110
EdgeTypeLoader, 110
Mode, 109
N_PHASES, 109
NotifyListeners, 110
ONE_PHASE, 109
Register, 110
Run, 111
SetAttributePositions, 111
SetAttributes, 111
SetFrequency, 111
SetGraph, 111
SetHeadAttribute, 112
SetHeadMEP, 112
SetHeadPosition, 112
SetLocale, 112
SetLogError, 112
SetLogOff, 113
SetRowReader, 113
SetTailAttribute, 113
SetTailMEP, 113
SetTailPosition, 114
SetTimestampFormat, 114
SetType, 114
TWO_PHASES, 109
Edges
Graph, 133
EdgesDirection
Gdb, 17
EnableType
DefaultExport, 87
ExportManager, 122
Encrypt
Sparksee, 267
EncryptedBackup
Graph, 134
EncryptionError, 114
EncryptionError, 116
Message, 116
SetMessage, 116
ThrowError, 117
Equal
Gdb, 16
Equals
Objects, 196
Value, 357
Error, 117
Error, 118
Message, 119
SetMessage, 119
ThrowError, 119
Exception, 119
Exception, 120
Message, 121
SetMessage, 121
ExcludeEdges
CommunitiesSCD, 54
CommunityDetection, 59
Connectivity, 66
Context, 70
DisjointCommunityDetection, 96
RandomWalk, 221
ShortestPath, 243
SinglePairShortestPath, 247
SinglePairShortestPathBFS, 253
SinglePairShortestPathDijkstra, 259
StrongConnectivity, 301
StrongConnectivityGabow, 307
Traversal, 317
TraversalBFS, 322
TraversalDFS, 327
WeakConnectivity, 388
WeakConnectivityDFS, 394
ExcludeNodes
CommunitiesSCD, 54
CommunityDetection, 59
Connectivity, 66
Context, 70
DisjointCommunityDetection, 96
RandomWalk, 221
ShortestPath, 243
SinglePairShortestPath, 248
SinglePairShortestPathBFS, 253
SinglePairShortestPathDijkstra, 259
StrongConnectivity, 301
StrongConnectivityGabow, 307
Traversal, 318
TraversalBFS, 322
TraversalDFS, 327
WeakConnectivity, 388
WeakConnectivityDFS, 394

Execute
Query, 211
Exists
Objects, 196
Explode
Graph, 134
Export
Graph, 134
ExportManager, 121
EnableType, 122
GetEdge, 122
GetEdgeType, 123
GetGraph, 123
GetNode, 123
GetNodeType, 124
Prepare, 124
Release, 124
ExportType
Gdb, 17

Fetch
QueryStream, 216
FileNotFoundException, 125
FileNotFoundException, 126
Message, 126
SetMessage, 126
FindAttribute
Graph, 135
FindAttributes
Graph, 135
FindEdge
Graph, 135
FindEdgeTypes
Graph, 136
FindNodeTypes
Graph, 136
FindObject
Graph, 136
FindOrCreateEdge
Graph, 136
FindOrCreateObject
Graph, 137
FindType
Graph, 137
FindTypes
Graph, 137
Fine
Gdb, 17

Gdb, 10
Any, 17
Ascendent, 18
AttributeKind, 15
Basic, 15
Between, 16
Boolean, 16
Box, 18
ColorRGB, 15
Condition, 15
Config, 17
Create, 18
DataType, 16
Debug, 17
Descendent, 18
Double, 16
Edge, 18
EdgesDirection, 17
Equal, 16
ExportType, 17
Fine, 17
GraphML, 17
Graphviz, 17
GreaterEqual, 16
GreaterThan, 16
Ignore, 18
Indexed, 15
Info, 17
Ingoing, 17
Integer, 16
InvalidChecksum, 19
IsError, 18
LessEqual, 16
LessThan, 16
Like, 16
LikeNoCase, 16
LogLevel, 17
Long, 16
MissingEndpoint, 17
Node, 18
NodeShape, 18
NotEqual, 16
OID, 17
ObjectType, 18
Off, 17
operator<<, 19
Order, 18
Outgoing, 17
QueryLanguage, 18
RegExp, 16
Round, 18
SPARKSEE_UNRECOVERABLE_ERR_CALLB↩
ACK, 15
Severe, 17
SparkseeAlgebra, 19
SparkseeCypher, 19
String, 16
SystemCallError, 19
Text, 16
Timestamp, 16
Unique, 15
UnrecoverableError, 19
Warning, 17
YGraphML, 17
GenerateSchemaScript
ScriptParser, 235
Get
ResultSetList, 228

SparkseeProperties, 293
ValueArray, 367
ValueList, 378
GetAESIV
SparkseeConfig, 277
GetAESKey
SparkseeConfig, 278
GetAlias
Database, 83
GetAreNeighborsIndexed
Type, 329
GetArrayAttribute
Graph, 137
GetArraySize
Attribute, 35
GetAttribute
Graph, 138
GetAttributeIntervalCount
Graph, 139
GetAttributeStatistics
Graph, 139
GetAttributeText
Graph, 140
GetAttributes
Graph, 139
GetAvailableMem
PlatformStatistics, 208
GetAvgLengthString
AttributeStatistics, 41
GetBoolean
SparkseeProperties, 293
Value, 357
ValueArray, 368
GetCache
DatabaseStatistics, 84
GetCacheMaxSize
Database, 83
SparkseeConfig, 278
GetCacheStatisticsEnabled
SparkseeConfig, 278
GetCacheStatisticsFile
SparkseeConfig, 278
GetCacheStatisticsSnapshotTime
SparkseeConfig, 278
GetCallStackDump
SparkseeConfig, 278
GetChecksumEnabled
SparkseeConfig, 279
GetClientId
SparkseeConfig, 279
GetColorRGB
EdgeExport, 100
NodeExport, 173
GetColumn
ResultSet, 225, 226
GetColumnDataType
ResultSet, 226
GetColumnIndex
ResultSet, 226
GetColumnName
ResultSet, 227
GetCommunities
CommunitiesSCD, 54
DisjointCommunityDetection, 96
GetCommunity
DisjointCommunities, 90
GetConnectedComponent
ConnectedComponents, 61
GetConnectedComponents
Connectivity, 66
StrongConnectivity, 302
StrongConnectivityGabow, 307
WeakConnectivity, 388
WeakConnectivityDFS, 394
GetCost
SinglePairShortestPath, 248
SinglePairShortestPathBFS, 254
SinglePairShortestPathDijkstra, 259
GetCount
Attribute, 35
ConnectedComponents, 62
DisjointCommunities, 91
Io, 23
TypeExporterEvent, 336
GetCurrentDepth
RandomWalk, 221
Traversal, 318
TraversalBFS, 323
TraversalDFS, 327
GetData
DatabaseStatistics, 84
GetDataType
Attribute, 35
Value, 357
GetDistinct
AttributeStatistics, 41
GetDouble
Value, 357
ValueArray, 368, 369
GetDownloadStatus
SparkseeConfig, 279
GetEdge
DefaultExport, 87
EdgeData, 98
ExportManager, 122
GetEdgeData
Graph, 140
GetEdgePeer
Graph, 140
GetEdgeType
DefaultExport, 88
ExportManager, 123
GetEncryptionEnabled
SparkseeConfig, 279
GetExtentPages
SparkseeConfig, 279

GetExtentSize
SparkseeConfig, 279
GetFontSize
EdgeExport, 100
NodeExport, 173
GetGraph
DefaultExport, 88
ExportManager, 123
Session, 239
GetHandler
Handler, 156
GetHead
EdgeData, 98
GetHeight
NodeExport, 173
GetHighAvailabilityCoordinators
SparkseeConfig, 280
GetHighAvailabilityEnabled
SparkseeConfig, 280
GetHighAvailabilityIP
SparkseeConfig, 280
GetHighAvailabilityMasterHistory
SparkseeConfig, 280
GetHighAvailabilitySynchronization
SparkseeConfig, 280
GetId
Attribute, 35
Type, 329
GetInMemAllocSize
SparkseeConfig, 280
GetInMemoryPoolCapacity
Session, 239
Sparksee, 268
GetInteger
SparkseeProperties, 294
Value, 358
ValueArray, 369
GetIsDirected
Type, 330
GetIsRestricted
Type, 330
GetJSON
ResultSet, 227
GetKind
Attribute, 36
GetLabel
EdgeExport, 100
GraphExport, 154
NodeExport, 173
GetLabelColorRGB
EdgeExport, 101
NodeExport, 173
GetLicense
SparkseeConfig, 281
GetLicenseId
SparkseeConfig, 281
GetLicensePreDownloadDays
SparkseeConfig, 281
GetLogFile
SparkseeConfig, 281
GetLogLevel
SparkseeConfig, 281
GetLong
Value, 358
ValueArray, 370
GetMax
AttributeStatistics, 41
GetMaxLengthString
AttributeStatistics, 41
GetMean
AttributeStatistics, 41
GetMedian
AttributeStatistics, 41
GetMin
AttributeStatistics, 42
GetMinLengthString
AttributeStatistics, 42
GetMode
AttributeStatistics, 42
GetModeCount
AttributeStatistics, 42
GetName
Attribute, 36
Type, 330
GetNode
DefaultExport, 88
ExportManager, 123
GetNodeType
DefaultExport, 89
ExportManager, 124
GetNodes
ConnectedComponents, 62
DisjointCommunities, 91
GetNull
AttributeStatistics, 42
GetNumCPUs
PlatformStatistics, 208
GetNumColumns
ResultSet, 227
GetNumObjects
Type, 330
GetOID
Value, 358
ValueArray, 371
GetObjectType
Graph, 140
Type, 330
GetPartition
Io, 23
GetPath
Database, 83
GetPathAsEdges
SinglePairShortestPath, 248
SinglePairShortestPathBFS, 254
SinglePairShortestPathDijkstra, 260
GetPathAsNodes

SinglePairShortestPath, 248
SinglePairShortestPathBFS, 254
SinglePairShortestPathDijkstra, 260
GetPhase
Io, 23
GetPoolFrameSize
SparkseeConfig, 281
GetPoolPartitions
SparkseeConfig, 282
GetPoolPersistentMaxSize
SparkseeConfig, 282
GetPoolPersistentMinSize
SparkseeConfig, 282
GetPoolTemporaryMaxSize
SparkseeConfig, 282
GetPoolTemporaryMinSize
SparkseeConfig, 282
GetRead
DatabaseStatistics, 85
GetRealTime
PlatformStatistics, 208
GetRecoveryCacheMaxSize
SparkseeConfig, 282
GetRecoveryCheckpointTime
SparkseeConfig, 283
GetRecoveryEnabled
SparkseeConfig, 283
GetRecoveryLogFile
SparkseeConfig, 283
GetRestrictedFrom
Type, 330
GetRestrictedTo
Type, 331
GetRollbackEnabled
SparkseeConfig, 283
GetRow
CSVReader, 75
RowReader, 231
GetSessions
DatabaseStatistics, 85
GetShape
NodeExport, 174
GetSize
Attribute, 36
ConnectedComponents, 62
DisjointCommunities, 91
GetSparkseeConfigFile
SparkseeConfig, 283
GetStatistics
Database, 83
Platform, 207
GetString
Value, 358
GetSystemTime
PlatformStatistics, 208
GetTail
EdgeData, 99
GetTemp
DatabaseStatistics, 85
GetTimeUnit
SparkseeProperties, 294
GetTimestamp
Value, 358
ValueArray, 371, 372
GetTmpEnabled
SparkseeConfig, 283
GetTmpFolder
SparkseeConfig, 284
GetTotal
AttributeStatistics, 43
TypeExporterEvent, 336
GetTotalMem
PlatformStatistics, 208
GetTotalPartitionSteps
Io, 23
GetTotalPartitions
Io, 23
GetTotalPhases
Io, 23
GetType
Graph, 141
GetTypeId
Attribute, 36
Io, 23
TypeExporterEvent, 337
GetUserTime
PlatformStatistics, 209
GetValues
Graph, 141
GetVariance
AttributeStatistics, 43
GetWidth
EdgeExport, 101
NodeExport, 174
GetWrite
DatabaseStatistics, 85
Graph, 127
Backup, 131
CountEdges, 132
CountNodes, 132
Degree, 132
Drop, 132, 133
DumpData, 133
DumpStorage, 133
Edges, 133
EncryptedBackup, 134
Explode, 134
Export, 134
FindAttribute, 135
FindAttributes, 135
FindEdge, 135
FindEdgeTypes, 136
FindNodeTypes, 136
FindObject, 136
FindOrCreateEdge, 136
FindOrCreateObject, 137

FindType, 137
FindTypes, 137
GetArrayAttribute, 137
GetAttribute, 138
GetAttributeIntervalCount, 139
GetAttributeStatistics, 139
GetAttributeText, 140
GetAttributes, 139
GetEdgeData, 140
GetEdgePeer, 140
GetObjectType, 140
GetType, 141
GetValues, 141
Heads, 141
IndexAttribute, 141
IndexNeighbors, 142
Neighbors, 142
NewArrayAttribute, 142
NewAttribute, 143
NewEdge, 143, 144
NewEdgeType, 144
NewNode, 144
NewNodeType, 145
NewRestrictedEdgeType, 145
NewSessionArrayAttribute, 145
NewSessionAttribute, 146
RemoveAttribute, 146
RemoveType, 147
RenameAttribute, 147
RenameType, 147
Select, 147–149
SetArrayAttribute, 149
SetArrayAttributeVoid, 149
SetAttribute, 150
SetAttributeDefaultValue, 150
SetAttributeText, 150
Tails, 151
TailsAndHeads, 151
TopK, 151–153
GraphExport, 153
GetLabel, 154
SetLabel, 154
GraphML
Gdb, 17
Graphviz
Gdb, 17
GreaterEqual
Gdb, 16
GreaterThan
Gdb, 16

Handler
~Handler, 156
GetHandler, 156
Handler, 156
IsNull, 156
SetHandler, 157
Handler< T >, 154
HasNext
AttributeListIterator, 39
BooleanListIterator, 45
Int32ListIterator, 159
KeyValues, 166
OIDListIterator, 202
ObjectsIterator, 199
RandomWalk, 221
ResultSetListIterator, 230
StringListIterator, 297
Traversal, 318
TraversalBFS, 323
TraversalDFS, 327
TypeListIterator, 341
ValueListIterator, 379
ValuesIterator, 383
Heads
Graph, 141

IOError, 160
IOError, 161
Message, 161
SetMessage, 162
ThrowError, 162
IOException, 162
IOException, 163
Message, 164
SetMessage, 164
Ignore
Gdb, 18
IncludeEdges
CommunitiesSCD, 54
CommunityDetection, 59
DisjointCommunityDetection, 96
IncludeNodes
CommunitiesSCD, 55
CommunityDetection, 59
DisjointCommunityDetection, 97
IndexAttribute
Graph, 141
IndexNeighbors
Graph, 142
Indexed
Gdb, 15
Info
Gdb, 17
Ingoing
Gdb, 17
Int32List, 157
Add, 158
Count, 158
Int32List, 158
Iterator, 158
Int32ListIterator, 159
HasNext, 159
Next, 159
Integer
Gdb, 16
Intersection
Objects, 197

InvalidChecksum
Gdb, 19
Io, 21
GetCount, 23
GetPartition, 23
GetPhase, 23
GetTotalPartitionSteps, 23
GetTotalPartitions, 23
GetTotalPhases, 23
GetTypeId, 23
IsLast, 24
operator<<, 24
IsArrayAttribute
Attribute, 36
IsError
Gdb, 18
IsFit
NodeExport, 174
IsLast
Io, 24
TypeExporterEvent, 337
IsNull
Handler, 156
TextStream, 314
Value, 359
IsSessionAttribute
Attribute, 36
Iterator
AttributeList, 38
BooleanList, 44
Int32List, 158
OIDList, 201
Objects, 197
ResultSetList, 229
StringList, 296
TypeList, 340
ValueList, 378
Values, 381
IteratorFromElement
Objects, 197
IteratorFromIndex
Objects, 197

KOpt, 167
Algorithms, 30
KeyValue, 164
KeyValues, 164
HasNext, 166
Next, 166

LessEqual
Gdb, 16
LessThan
Gdb, 16
LicenseError, 168
LicenseError, 170
Message, 170
SetMessage, 170
ThrowError, 170
Like
Gdb, 16
LikeNoCase
Gdb, 16
Load
SparkseeProperties, 294
LoadEdges
CSVLoader, 71
LoadNodes
CSVLoader, 72
LogLevel
Gdb, 17
Long
Gdb, 16

MakeNull
ValueArray, 372
maxHops
ShortestPath, 244
SinglePairShortestPath, 249
SinglePairShortestPathBFS, 255
SinglePairShortestPathDijkstra, 261
Message
AppError, 33
EncryptionError, 116
Error, 119
Exception, 121
FileNotFoundException, 126
IOError, 161
IOException, 164
LicenseError, 170
NoSuchElementException, 189
QueryException, 215
SystemError, 311
UnsupportedOperationError, 352
WrongArgumentError, 397
MissingEndpoint
Gdb, 17
Mode
EdgeTypeLoader, 109
NodeTypeLoader, 183
TypeLoader, 343

N_PHASES
EdgeTypeLoader, 109
NodeTypeLoader, 183
TypeLoader, 343
Neighbors
Graph, 142
NewArrayAttribute
Graph, 142
NewAttribute
Graph, 143
NewEdge
Graph, 143, 144
NewEdgeType
Graph, 144
NewNode
Graph, 144

NewNodeType
Graph, 145
NewObjects
Session, 239
NewQuery
Session, 239
NewRestrictedEdgeType
Graph, 145
NewSessionArrayAttribute
Graph, 145
NewSessionAttribute
Graph, 146
Next
AttributeListIterator, 39
BooleanListIterator, 45
Int32ListIterator, 159
KeyValues, 166
OIDListIterator, 202
ObjectsIterator, 199
RandomWalk, 222
ResultSet, 227
ResultSetListIterator, 230
StringListIterator, 297
Traversal, 318
TraversalBFS, 323
TraversalDFS, 328
TypeListIterator, 341
ValueListIterator, 379
ValuesIterator, 383
NoSuchElementException, 188
Message, 189
NoSuchElementException, 189
SetMessage, 189
Node
Gdb, 18
NodeExport, 171
GetColorRGB, 173
GetFontSize, 173
GetHeight, 173
GetLabel, 173
GetLabelColorRGB, 173
GetShape, 174
GetWidth, 174
IsFit, 174
SetColorRGB, 174
SetFit, 174
SetFontSize, 175
SetHeight, 175
SetLabel, 175
SetLabelColorRGB, 175
SetShape, 175
SetWidth, 175
NodeShape
Gdb, 18
NodeTypeExporter, 176
CanRun, 178
NodeTypeExporter, 177
NotifyListeners, 178
Register, 178
RunProcess, 178
SetAttributes, 178
SetFrequency, 179
SetGraph, 179
SetHeadAttribute, 179
SetHeadPosition, 179
SetHeader, 179
SetRowWriter, 180
SetTailAttribute, 180
SetTailPosition, 180
SetType, 180
NodeTypeLoader, 181
CanRun, 183
Mode, 183
N_PHASES, 183
NodeTypeLoader, 183
NotifyListeners, 184
ONE_PHASE, 183
Register, 184
Run, 184
SetAttributePositions, 184
SetAttributes, 184
SetFrequency, 185
SetGraph, 185
SetHeadAttribute, 185
SetHeadMEP, 185
SetHeadPosition, 185
SetLocale, 186
SetLogError, 186
SetLogOff, 186
SetRowReader, 186
SetTailAttribute, 187
SetTailMEP, 187
SetTailPosition, 187
SetTimestampFormat, 187
SetType, 187
TWO_PHASES, 183
NotEqual
Gdb, 16
NotifyEvent
TypeExporterListener, 338
TypeLoaderListener, 350
NotifyListeners
EdgeTypeExporter, 105
EdgeTypeLoader, 110
NodeTypeExporter, 178
NodeTypeLoader, 184
TypeExporter, 333
TypeLoader, 344

OIDList, 200
Add, 201
Count, 201
Iterator, 201
OIDList, 200, 201
Set, 201
OIDListIterator, 202
HasNext, 202

Next, 202
OID
Gdb, 17
ONE_PHASE
EdgeTypeLoader, 109
NodeTypeLoader, 183
TypeLoader, 343
ObjectType
Gdb, 18
Objects, 191
Add, 193
Any, 194
CombineDifference, 194
CombineIntersection, 194
CombineUnion, 195
Contains, 195
Copy, 195
Count, 196
Difference, 196
Equals, 196
Exists, 196
Intersection, 197
Iterator, 197
IteratorFromElement, 197
IteratorFromIndex, 197
Remove, 198
Sample, 198
Union, 198
ObjectsIterator, 199
HasNext, 199
Next, 199
Off
Gdb, 17
Open
CSVReader, 75
CSVWriter, 79
Sparksee, 268
operator<<
Gdb, 19
Io, 24
Script, 25
operator=
Value, 359
Order
Gdb, 18
Outgoing
Gdb, 17

PageRank, 203
AddAllEdgeTypes, 204
AddEdgeType, 204
PageRank, 204
Run, 204
SetDamping, 205
SetDefaultWeight, 205
SetEdgeWeightAttributeType, 205
SetInitialPageRankValue, 205
SetNumIterations, 206
SetOutputAttributeType, 206
SetStartingNode, 206
SetTolerance, 206
Parse
ScriptParser, 235, 236
pathAsEdges
SinglePairShortestPath, 249
SinglePairShortestPathBFS, 255
SinglePairShortestPathDijkstra, 261
Platform, 206
GetStatistics, 207
PlatformStatistics, 207
GetAvailableMem, 208
GetNumCPUs, 208
GetRealTime, 208
GetSystemTime, 208
GetTotalMem, 208
GetUserTime, 209
PreCommit
Session, 240
Prepare
ExportManager, 124
QueryStream, 217

Query, 209
Execute, 211
SetDynamic, 211
SetStream, 211
QueryContext, 212
QueryException, 212
Message, 215
QueryException, 214, 215
SetMessage, 215
ThrowError, 215
QueryLanguage
Gdb, 18
QueryStream, 216
Fetch, 216
Prepare, 217
Start, 217

RandomWalk, 217
AddAllEdgeTypes, 220
AddEdgeType, 220
AddNodeType, 220
ExcludeEdges, 221
ExcludeNodes, 221
GetCurrentDepth, 221
HasNext, 221
Next, 222
RandomWalk, 220
Reset, 222
SetDefaultWeight, 222
SetEdgeWeightAttributeType, 222
SetInOutParameter, 223
SetMaximumHops, 223
SetReturnParameter, 223
SetSeed, 223
Read
CSVReader, 75

RowReader, 232
TextStream, 314
RedoPrecommitted
Database, 83
RegExp
Gdb, 16
Register
EdgeTypeExporter, 105
EdgeTypeLoader, 110
NodeTypeExporter, 178
NodeTypeLoader, 184
TypeExporter, 333
TypeLoader, 344
Release
ExportManager, 124
Remove
Objects, 198
RemoveAttribute
Graph, 146
RemoveChecksums
Sparksee, 269
RemoveType
Graph, 147
RenameAttribute
Graph, 147
RenameType
Graph, 147
Reset
CSVReader, 76
RandomWalk, 222
RowReader, 232
ResizeInMemoryPool
Sparksee, 269
Restore
Sparksee, 269, 270
RestoreEncryptedBackup
Sparksee, 270, 271
ResultSet, 224
GetColumn, 225, 226
GetColumnDataType, 226
GetColumnIndex, 226
GetColumnName, 227
GetJSON, 227
GetNumColumns, 227
Next, 227
ResultSetList, 228
Count, 228
Get, 228
Iterator, 229
ResultSetList, 228
ResultSetListIterator, 229
HasNext, 230
Next, 230
Round
Gdb, 18
RowReader, 230
Close, 231
GetRow, 231
Read, 232
Reset, 232
RowReader, 231
RowWriter, 232
Close, 234
RowWriter, 233
Write, 234
Run
CommunityDetection, 60
Connectivity, 67
DisjointCommunityDetection, 97
EdgeTypeLoader, 111
NodeTypeLoader, 184
PageRank, 204
ShortestPath, 243
SinglePairShortestPath, 248
StrongConnectivity, 302
TypeExporter, 333
TypeLoader, 344
WeakConnectivity, 389
RunNPhases
TypeLoader, 345
RunProcess
EdgeTypeExporter, 105
NodeTypeExporter, 178
TypeExporter, 333
RunTwoPhases
TypeLoader, 345

SPARKSEE_UNRECOVERABLE_ERR_CALLBACK
Gdb, 15
Sample
Objects, 198
Save
SparkseeConfig, 284
SaveAll
SparkseeConfig, 284
Script, 25
operator<<, 25
ScriptParser, 234
GenerateSchemaScript, 235
Parse, 235, 236
SetErrorLog, 236
SetOutputLog, 236
Select
Graph, 147–149
Session, 237
GetGraph, 239
GetInMemoryPoolCapacity, 239
NewObjects, 239
NewQuery, 239
PreCommit, 240
Set
OIDList, 201
Value, 359
ValueArray, 372
SetAESEncryptionEnabled
SparkseeConfig, 284, 285
SetArrayAttribute

Graph, 149
SetArrayAttributeVoid
Graph, 149
SetAsDirected
EdgeExport, 101
SetAttribute
Graph, 150
SetAttributeDefaultValue
Graph, 150
SetAttributePositions
BulkLoader, 47
EdgeTypeLoader, 111
NodeTypeLoader, 184
TypeLoader, 345
SetAttributeText
Graph, 150
SetAttributes
BulkLoader, 47
EdgeTypeExporter, 105
EdgeTypeLoader, 111
NodeTypeExporter, 178
NodeTypeLoader, 184
TypeExporter, 334
TypeLoader, 345
SetAutoQuotes
CSVWriter, 79
SetBoolean
Value, 359
ValueArray, 373
SetBooleanVoid
Value, 360
SetCacheMaxSize
Database, 83
SparkseeConfig, 285
SetCacheStatisticsEnabled
SparkseeConfig, 285
SetCacheStatisticsFile
SparkseeConfig, 285
SetCacheStatisticsSnapshotTime
SparkseeConfig, 285
SetCallStackDump
SparkseeConfig, 287
SetChecksumEnabled
SparkseeConfig, 287
SetClientId
SparkseeConfig, 287
SetColorRGB
EdgeExport, 101
NodeExport, 174
SetCurrentTour
Algorithms, 30
SetDamping
PageRank, 205
SetDefaultWeight
PageRank, 205
RandomWalk, 222
SetDouble
Value, 360
ValueArray, 373, 374
SetDoubleVoid
Value, 360
SetDynamic
Query, 211
SetDynamicEdgeCostCallback
SinglePairShortestPathDijkstra, 260
SetEdgeWeightAttributeType
Algorithms, 31
PageRank, 205
RandomWalk, 222
SetErrorLog
ScriptParser, 236
SetExtentPages
SparkseeConfig, 287
SetExtentSize
SparkseeConfig, 287
SetFilename
BulkLoader, 48
SetFit
NodeExport, 174
SetFontSize
EdgeExport, 101
NodeExport, 175
SetForcedQuotes
CSVWriter, 79
SetFrequency
EdgeTypeExporter, 105
EdgeTypeLoader, 111
NodeTypeExporter, 179
NodeTypeLoader, 185
TypeExporter, 334
TypeLoader, 345
SetGraph
EdgeTypeExporter, 105
EdgeTypeLoader, 111
NodeTypeExporter, 179
NodeTypeLoader, 185
TypeExporter, 334
TypeLoader, 346
SetHandler
Handler, 157
SetHeadAttribute
EdgeTypeExporter, 106
EdgeTypeLoader, 112
NodeTypeExporter, 179
NodeTypeLoader, 185
TypeExporter, 334
TypeLoader, 346
SetHeadMEP
EdgeTypeLoader, 112
NodeTypeLoader, 185
TypeLoader, 346
SetHeadPosition
EdgeTypeExporter, 106
EdgeTypeLoader, 112
NodeTypeExporter, 179
NodeTypeLoader, 185

TypeExporter, 335
TypeLoader, 346
SetHeader
EdgeTypeExporter, 106
NodeTypeExporter, 179
TypeExporter, 334
SetHeight
NodeExport, 175
SetHI
SparkseeProperties, 295
SetHighAvailabilityCoordinators
SparkseeConfig, 288
SetHighAvailabilityEnabled
SparkseeConfig, 288
SetHighAvailabilityIP
SparkseeConfig, 288
SetHighAvailabilityMasterHistory
SparkseeConfig, 288
SetHighAvailabilitySynchronization
SparkseeConfig, 288
SetInMemAllocSize
SparkseeConfig, 288
SetInOutParameter
RandomWalk, 223
SetInitialPageRankValue
PageRank, 205
SetInteger
Value, 360
ValueArray, 374
SetIntegerVoid
Value, 360
SetLabel
EdgeExport, 102
GraphExport, 154
NodeExport, 175
SetLabelColorRGB
EdgeExport, 102
NodeExport, 175
SetLicense
SparkseeConfig, 289
SetLicenseId
SparkseeConfig, 289
SetLicensePreDownloadDays
SparkseeConfig, 289
SetLocale
BulkLoader, 48
CSVReader, 76
CSVWriter, 79
EdgeTypeLoader, 112
NodeTypeLoader, 186
TypeLoader, 346
SetLogError
BulkLoader, 48
EdgeTypeLoader, 112
NodeTypeLoader, 186
TypeLoader, 347
SetLogFile
SparkseeConfig, 289
SetLogLevel
SparkseeConfig, 289
SetLogOff
EdgeTypeLoader, 113
NodeTypeLoader, 186
TypeLoader, 347
SetLong
Value, 361
ValueArray, 375
SetLongVoid
Value, 361
SetLookAhead
CommunitiesSCD, 55
SetMaterializedAttribute
CommunitiesSCD, 55
Connectivity, 67
DisjointCommunityDetection, 97
StrongConnectivity, 302
StrongConnectivityGabow, 307
WeakConnectivity, 389
WeakConnectivityDFS, 395
SetMaxIterations
Algorithms, 31
SetMaximumHops
Context, 70
RandomWalk, 223
ShortestPath, 243
SinglePairShortestPath, 249
SinglePairShortestPathBFS, 254
SinglePairShortestPathDijkstra, 260
Traversal, 318
TraversalBFS, 323
TraversalDFS, 328
SetMessage
AppError, 33
EncryptionError, 116
Error, 119
Exception, 121
FileNotFoundException, 126
IOError, 162
IOException, 164
LicenseError, 170
NoSuchElementException, 189
QueryException, 215
SystemError, 311
UnsupportedOperationError, 352
WrongArgumentError, 397
SetMultilines
CSVReader, 76
SetNodesNotVisited
CommunitiesSCD, 55
CommunityDetection, 60
Connectivity, 67
DisjointCommunityDetection, 98
StrongConnectivity, 302
StrongConnectivityGabow, 308
WeakConnectivity, 389
WeakConnectivityDFS, 395

SetNull
Value, 361
SetNumIterations
PageRank, 206
SetNumLines
CSVReader, 76
SetOIDVoid
Value, 361
SetOID
Value, 361
ValueArray, 375, 376
SetOutputAttributeType
PageRank, 206
SetOutputLog
ScriptParser, 236
SetPoolFrameSize
SparkseeConfig, 290
SetPoolPartitions
SparkseeConfig, 290
SetPoolPersistentMaxSize
SparkseeConfig, 290
SetPoolPersistentMinSize
SparkseeConfig, 290
SetPoolTemporaryMaxSize
SparkseeConfig, 290
SetPoolTemporaryMinSize
SparkseeConfig, 290
SetQuotes
CSVReader, 76
CSVWriter, 80
SetRecoveryCacheMaxSize
SparkseeConfig, 291
SetRecoveryCheckpointTime
SparkseeConfig, 291
SetRecoveryEnabled
SparkseeConfig, 291
SetRecoveryLogFile
SparkseeConfig, 291
SetReturnParameter
RandomWalk, 223
SetRollbackEnabled
SparkseeConfig, 291
SetRowReader
EdgeTypeLoader, 113
NodeTypeLoader, 186
TypeLoader, 347
SetRowWriter
EdgeTypeExporter, 106
NodeTypeExporter, 180
TypeExporter, 335
SetSeed
RandomWalk, 223
SetSeparator
CSVReader, 77
CSVWriter, 80
SetShape
NodeExport, 175
SetSparkseeConfigFile
SparkseeConfig, 292
SetStartLine
CSVReader, 77
SetStartingNode
PageRank, 206
SetStream
Query, 211
SetString
Value, 362
SetStringVoid
Value, 362
SetTailAttribute
EdgeTypeExporter, 106
EdgeTypeLoader, 113
NodeTypeExporter, 180
NodeTypeLoader, 187
TypeExporter, 335
TypeLoader, 347
SetTailMEP
EdgeTypeLoader, 113
NodeTypeLoader, 187
TypeLoader, 347
SetTailPosition
EdgeTypeExporter, 107
EdgeTypeLoader, 114
NodeTypeExporter, 180
NodeTypeLoader, 187
TypeExporter, 335
TypeLoader, 348
SetTimeLimit
Algorithms, 31
SetTimestamp
Value, 362
ValueArray, 376
SetTimestampFormat
BulkLoader, 48
EdgeTypeLoader, 114
NodeTypeLoader, 187
TypeLoader, 348
SetTimestampVoid
Value, 363
SetTmpEnabled
SparkseeConfig, 292
SetTmpFolder
SparkseeConfig, 292
SetTolerance
PageRank, 206
SetType
BulkLoader, 48
EdgeTypeExporter, 107
EdgeTypeLoader, 114
NodeTypeExporter, 180
NodeTypeLoader, 187
TypeExporter, 335
TypeLoader, 348
SetUnrecoverableErrorCallback
Sparksee, 271
SetUnweightedEdgeCost

SinglePairShortestPathDijkstra, 260
SetVoid
Value, 363
SetWidth
EdgeExport, 102
NodeExport, 175
Severe
Gdb, 17
ShortestPath, 240
AddAllEdgeTypes, 242
AddEdgeType, 243
ExcludeEdges, 243
ExcludeNodes, 243
maxHops, 244
Run, 243
SetMaximumHops, 243
ShortestPath, 242
SinglePairShortestPath, 244
AddAllEdgeTypes, 247
AddEdgeType, 247
ExcludeEdges, 247
ExcludeNodes, 248
GetCost, 248
GetPathAsEdges, 248
GetPathAsNodes, 248
maxHops, 249
pathAsEdges, 249
Run, 248
SetMaximumHops, 249
SinglePairShortestPath, 247
SinglePairShortestPathBFS, 249
AddAllEdgeTypes, 253
AddEdgeType, 253
CheckOnlyExistence, 253
ExcludeEdges, 253
ExcludeNodes, 253
GetCost, 254
GetPathAsEdges, 254
GetPathAsNodes, 254
maxHops, 255
pathAsEdges, 255
SetMaximumHops, 254
SinglePairShortestPathBFS, 252
SinglePairShortestPathDijkstra, 255
AddAllEdgeTypes, 258
AddEdgeType, 259
AddWeightedEdgeType, 259
ExcludeEdges, 259
ExcludeNodes, 259
GetCost, 259
GetPathAsEdges, 260
GetPathAsNodes, 260
maxHops, 261
pathAsEdges, 261
SetDynamicEdgeCostCallback, 260
SetMaximumHops, 260
SetUnweightedEdgeCost, 260
SinglePairShortestPathDijkstra, 258
SinglePairShortestPathDijkstra::FibonacciHeap::Node, 171
SinglePairShortestPathDijkstraDynamicCost, 261
CalculateEdgeCost, 262
Sparksee, 262
AddChecksums, 265
Create, 266
Decrypt, 267
Encrypt, 267
GetInMemoryPoolCapacity, 268
Open, 268
RemoveChecksums, 269
ResizeInMemoryPool, 269
Restore, 269, 270
RestoreEncryptedBackup, 270, 271
SetUnrecoverableErrorCallback, 271
Sparksee, 265
VerifyChecksums, 271
SparkseeAlgebra
Gdb, 19
SparkseeConfig, 272
DownloadLicense, 277
GetAESIV, 277
GetAESKey, 278
GetCacheMaxSize, 278
GetCacheStatisticsEnabled, 278
GetCacheStatisticsFile, 278
GetCacheStatisticsSnapshotTime, 278
GetCallStackDump, 278
GetChecksumEnabled, 279
GetClientId, 279
GetDownloadStatus, 279
GetEncryptionEnabled, 279
GetExtentPages, 279
GetExtentSize, 279
GetHighAvailabilityCoordinators, 280
GetHighAvailabilityEnabled, 280
GetHighAvailabilityIP, 280
GetHighAvailabilityMasterHistory, 280
GetHighAvailabilitySynchronization, 280
GetInMemAllocSize, 280
GetLicense, 281
GetLicenseId, 281
GetLicensePreDownloadDays, 281
GetLogFile, 281
GetLogLevel, 281
GetPoolFrameSize, 281
GetPoolPartitions, 282
GetPoolPersistentMaxSize, 282
GetPoolPersistentMinSize, 282
GetPoolTemporaryMaxSize, 282
GetPoolTemporaryMinSize, 282
GetRecoveryCacheMaxSize, 282
GetRecoveryCheckpointTime, 283
GetRecoveryEnabled, 283
GetRecoveryLogFile, 283
GetRollbackEnabled, 283
GetSparkseeConfigFile, 283

GetTmpEnabled, 283
GetTmpFolder, 284
Save, 284
SaveAll, 284
SetAESEncryptionEnabled, 284, 285
SetCacheMaxSize, 285
SetCacheStatisticsEnabled, 285
SetCacheStatisticsFile, 285
SetCacheStatisticsSnapshotTime, 285
SetCallStackDump, 287
SetChecksumEnabled, 287
SetClientId, 287
SetExtentPages, 287
SetExtentSize, 287
SetHighAvailabilityCoordinators, 288
SetHighAvailabilityEnabled, 288
SetHighAvailabilityIP, 288
SetHighAvailabilityMasterHistory, 288
SetHighAvailabilitySynchronization, 288
SetInMemAllocSize, 288
SetLicense, 289
SetLicenseId, 289
SetLicensePreDownloadDays, 289
SetLogFile, 289
SetLogLevel, 289
SetPoolFrameSize, 290
SetPoolPartitions, 290
SetPoolPersistentMaxSize, 290
SetPoolPersistentMinSize, 290
SetPoolTemporaryMaxSize, 290
SetPoolTemporaryMinSize, 290
SetRecoveryCacheMaxSize, 291
SetRecoveryCheckpointTime, 291
SetRecoveryEnabled, 291
SetRecoveryLogFile, 291
SetRollbackEnabled, 291
SetSparkseeConfigFile, 292
SetTmpEnabled, 292
SetTmpFolder, 292
SparkseeConfig, 277
SparkseeCypher
Gdb, 19
SparkseeProperties, 292
Get, 293
GetBoolean, 293
GetInteger, 294
GetTimeUnit, 294
Load, 294
SetHI, 295
Start
QueryStream, 217
String
Gdb, 16
StringList, 295
Add, 296
Count, 296
Iterator, 296
StringList, 295
StringListIterator, 296
HasNext, 297
Next, 297
StrongConnectivity, 298
AddAllEdgeTypes, 301
AddEdgeType, 301
ExcludeEdges, 301
ExcludeNodes, 301
GetConnectedComponents, 302
Run, 302
SetMaterializedAttribute, 302
SetNodesNotVisited, 302
StrongConnectivity, 301
StrongConnectivityGabow, 303
AddAllEdgeTypes, 306
AddEdgeType, 306
ExcludeEdges, 307
ExcludeNodes, 307
GetConnectedComponents, 307
SetMaterializedAttribute, 307
SetNodesNotVisited, 308
StrongConnectivityGabow, 306
SystemCallError
Gdb, 19
SystemError, 308
Message, 311
SetMessage, 311
SystemError, 310
ThrowError, 311

TWO_PHASES
EdgeTypeLoader, 109
NodeTypeLoader, 183
TypeLoader, 343
Tails
Graph, 151
TailsAndHeads
Graph, 151
Text
Gdb, 16
TextStream, 311
Close, 314
IsNull, 314
Read, 314
TextStream, 313
Write, 314
ThrowError
AppError, 34
EncryptionError, 117
Error, 119
IOError, 162
LicenseError, 170
QueryException, 215
SystemError, 311
UnsupportedOperationError, 353
WrongArgumentError, 398
Timestamp
Gdb, 16
ToString

Value, 363
TopK
Graph, 151–153
Traversal, 315
AddAllEdgeTypes, 317
AddEdgeType, 317
AddNodeType, 317
ExcludeEdges, 317
ExcludeNodes, 318
GetCurrentDepth, 318
HasNext, 318
Next, 318
SetMaximumHops, 318
Traversal, 316
TraversalBFS, 319
AddAllEdgeTypes, 321
AddEdgeType, 322
AddNodeType, 322
ExcludeEdges, 322
ExcludeNodes, 322
GetCurrentDepth, 323
HasNext, 323
Next, 323
SetMaximumHops, 323
TraversalBFS, 321
TraversalDFS, 324
AddAllEdgeTypes, 326
AddEdgeType, 326
AddNodeType, 326
ExcludeEdges, 327
ExcludeNodes, 327
GetCurrentDepth, 327
HasNext, 327
Next, 328
SetMaximumHops, 328
TraversalDFS, 326
Type, 328
GetAreNeighborsIndexed, 329
GetId, 329
GetIsDirected, 330
GetIsRestricted, 330
GetName, 330
GetNumObjects, 330
GetObjectType, 330
GetRestrictedFrom, 330
GetRestrictedTo, 331
TypeExporter, 331
CanRun, 333
NotifyListeners, 333
Register, 333
Run, 333
RunProcess, 333
SetAttributes, 334
SetFrequency, 334
SetGraph, 334
SetHeadAttribute, 334
SetHeadPosition, 335
SetHeader, 334
SetRowWriter, 335
SetTailAttribute, 335
SetTailPosition, 335
SetType, 335
TypeExporter, 332
TypeExporterEvent, 336
GetCount, 336
GetTotal, 336
GetTypeId, 337
IsLast, 337
TypeExporterListener, 337
NotifyEvent, 338
TypeExporterListener, 338
TypeList, 338
Add, 339
Count, 339
Iterator, 340
TypeList, 339
TypeListIterator, 340
HasNext, 341
Next, 341
TypeLoader, 341
CanRun, 344
Mode, 343
N_PHASES, 343
NotifyListeners, 344
ONE_PHASE, 343
Register, 344
Run, 344
RunNPhases, 345
RunTwoPhases, 345
SetAttributePositions, 345
SetAttributes, 345
SetFrequency, 345
SetGraph, 346
SetHeadAttribute, 346
SetHeadMEP, 346
SetHeadPosition, 346
SetLocale, 346
SetLogError, 347
SetLogOff, 347
SetRowReader, 347
SetTailAttribute, 347
SetTailMEP, 347
SetTailPosition, 348
SetTimestampFormat, 348
SetType, 348
TWO_PHASES, 343
TypeLoader, 343
TypeLoaderEvent, 348
TypeLoaderListener, 349
NotifyEvent, 350
TypeLoaderListener, 350

Union
Objects, 198
Unique
Gdb, 15
UnrecoverableError

Gdb, 19
UnsupportedOperationError, 350
Message, 352
SetMessage, 352
ThrowError, 353
UnsupportedOperationError, 352

Value, 353
Compare, 357
Equals, 357
GetBoolean, 357
GetDataType, 357
GetDouble, 357
GetInteger, 358
GetLong, 358
GetOID, 358
GetString, 358
GetTimestamp, 358
IsNull, 359
operator=, 359
Set, 359
SetBoolean, 359
SetBooleanVoid, 360
SetDouble, 360
SetDoubleVoid, 360
SetInteger, 360
SetIntegerVoid, 360
SetLong, 361
SetLongVoid, 361
SetNull, 361
SetOIDVoid, 361
SetOID, 361
SetString, 362
SetStringVoid, 362
SetTimestamp, 362
SetTimestampVoid, 363
SetVoid, 363
ToString, 363
Value, 356
ValueArray, 364
Get, 367
GetBoolean, 368
GetDouble, 368, 369
GetInteger, 369
GetLong, 370
GetOID, 371
GetTimestamp, 371, 372
MakeNull, 372
Set, 372
SetBoolean, 373
SetDouble, 373, 374
SetInteger, 374
SetLong, 375
SetOID, 375, 376
SetTimestamp, 376
ValueList, 377
Add, 377
Count, 378
Get, 378
Iterator, 378
ValueList, 377
ValueListIterator, 378
HasNext, 379
Next, 379
Values, 380
Count, 381
Iterator, 381
ValuesIterator, 382
HasNext, 383
Next, 383
VerifyChecksums
Sparksee, 271

Warning
Gdb, 17
WeakConnectivity, 384
AddAllEdgeTypes, 387
AddEdgeType, 387, 388
ExcludeEdges, 388
ExcludeNodes, 388
GetConnectedComponents, 388
Run, 389
SetMaterializedAttribute, 389
SetNodesNotVisited, 389
WeakConnectivity, 387
WeakConnectivityDFS, 389
AddAllEdgeTypes, 393
AddEdgeType, 394
ExcludeEdges, 394
ExcludeNodes, 394
GetConnectedComponents, 394
SetMaterializedAttribute, 395
SetNodesNotVisited, 395
WeakConnectivityDFS, 393
Write
CSVWriter, 80
RowWriter, 234
TextStream, 314
WrongArgumentError, 395
Message, 397
SetMessage, 397
ThrowError, 398
WrongArgumentError, 397

YGraphML
Gdb, 17