

SpatiaLite

5.1.0

Generated by Doxygen 1.9.1

1 Introduction	1
1.1 Generalities	1
1.2 Building	2
1.3 Deployment	3
1.4 License	3
2 Deprecated List	5
3 Data Structure Index	7
3.1 Data Structures	7
4 File Index	9
4.1 File List	9
5 Data Structure Documentation	11
5.1 gaia_dxf_arc Struct Reference	11
5.1.1 Detailed Description	11
5.2 gaia_dxf_block Struct Reference	12
5.2.1 Detailed Description	13
5.3 gaia_dxf_boundary_path Struct Reference	13
5.3.1 Detailed Description	14
5.4 gaia_dxf_circle Struct Reference	14
5.4.1 Detailed Description	14
5.5 gaia_dxf_extra_attr Struct Reference	15
5.5.1 Detailed Description	15
5.6 gaia_dxf_hatch Struct Reference	15
5.6.1 Detailed Description	16
5.7 gaia_dxf_hatch_segm Struct Reference	16
5.7.1 Detailed Description	17
5.8 gaia_dxf_hole Struct Reference	17
5.8.1 Detailed Description	18
5.9 gaia_dxf_insert Struct Reference	18
5.9.1 Detailed Description	19
5.10 gaia_dxf_layer Struct Reference	19
5.10.1 Detailed Description	22
5.11 gaia_dxf_parser Struct Reference	22
5.11.1 Detailed Description	24
5.12 gaia_dxf_point Struct Reference	25
5.12.1 Detailed Description	25
5.13 gaia_dxf_polyline Struct Reference	26
5.13.1 Detailed Description	26
5.14 gaia_dxf_text Struct Reference	27
5.14.1 Detailed Description	27
5.15 gaia_dxf_write Struct Reference	28

5.15.1 Detailed Description	28
5.16 gaiaAttributeFieldDoubleRangeInfos Struct Reference	28
5.16.1 Detailed Description	28
5.17 gaiaAttributeFieldIntRangeInfos Struct Reference	29
5.17.1 Detailed Description	29
5.18 gaiaAttributeFieldMaxSizeInfos Struct Reference	29
5.18.1 Detailed Description	29
5.19 gaiaDbfFieldStruct Struct Reference	30
5.19.1 Detailed Description	30
5.20 gaiaDbfListStruct Struct Reference	31
5.20.1 Detailed Description	31
5.21 gaiaDbfStruct Struct Reference	32
5.21.1 Detailed Description	33
5.22 gaiaDynamicLineStruct Struct Reference	33
5.22.1 Detailed Description	33
5.23 gaiaExifTagListStruct Struct Reference	34
5.23.1 Detailed Description	34
5.24 gaiaExifTagStruct Struct Reference	35
5.24.1 Detailed Description	36
5.25 gaiaGeomCollStruct Struct Reference	36
5.25.1 Detailed Description	37
5.26 gaiaLayerAttributeFieldInfos Struct Reference	37
5.26.1 Detailed Description	38
5.27 gaiaLayerAuthInfos Struct Reference	38
5.27.1 Detailed Description	39
5.28 gaiaLayerExtentInfos Struct Reference	39
5.28.1 Detailed Description	39
5.29 gaiaLinestringStruct Struct Reference	40
5.29.1 Detailed Description	40
5.30 gaiaMemFileStruct Struct Reference	41
5.30.1 Detailed Description	41
5.31 gaiaOutBufferStruct Struct Reference	41
5.31.1 Detailed Description	41
5.32 gaiaPointStruct Struct Reference	42
5.32.1 Detailed Description	42
5.33 gaiaPolygonStruct Struct Reference	43
5.33.1 Detailed Description	44
5.34 gaiaPreRingStruct Struct Reference	44
5.34.1 Detailed Description	44
5.35 gaiaProjAreaStr Struct Reference	45
5.35.1 Detailed Description	45
5.36 gaiaRingStruct Struct Reference	45

5.36.1 Detailed Description	46
5.37 gaiaShapefileStruct Struct Reference	46
5.37.1 Detailed Description	48
5.38 gaiaValueStruct Struct Reference	48
5.38.1 Detailed Description	49
5.39 gaiaVectorLayerItem Struct Reference	49
5.39.1 Detailed Description	50
5.40 gaiaVectorLayersListStr Struct Reference	50
5.40.1 Detailed Description	50
5.41 vrttxt_column_header Struct Reference	51
5.41.1 Detailed Description	51
5.42 vrttxt_line Struct Reference	51
5.42.1 Detailed Description	51
5.43 vrttxt_reader Struct Reference	52
5.43.1 Detailed Description	53
5.44 vrttxt_row Struct Reference	53
5.44.1 Detailed Description	54
5.45 vrttxt_row_block Struct Reference	54
5.45.1 Detailed Description	54
6 File Documentation	55
6.1 src/headers/spatialite.h File Reference	55
6.1.1 Detailed Description	60
6.1.2 Function Documentation	60
6.1.2.1 check_all_geometry_columns()	60
6.1.2.2 check_all_geometry_columns_r()	61
6.1.2.3 check_duplicated_rows()	62
6.1.2.4 check_geometry_column()	62
6.1.2.5 check_geometry_column_r()	63
6.1.2.6 disable_tiny_point()	64
6.1.2.7 dump_dbf()	64
6.1.2.8 dump_dbf_ex()	65
6.1.2.9 dump_dbf_ex2()	65
6.1.2.10 dump_geojson()	66
6.1.2.11 dump_geojson2()	67
6.1.2.12 dump_geojson_ex()	68
6.1.2.13 dump_kml()	69
6.1.2.14 dump_kml_ex()	69
6.1.2.15 dump_shapefile()	70
6.1.2.16 dump_shapefile_ex()	71
6.1.2.17 dump_shapefile_ex2()	72
6.1.2.18 elementary_geometries()	72

6.1.2.19 elementary_geometries_ex()	73
6.1.2.20 elementary_geometries_ex2()	74
6.1.2.21 elementary_geometries_ex3()	74
6.1.2.22 enable_tiny_point()	75
6.1.2.23 gaia_create_routing()	76
6.1.2.24 gaia_create_routing_nodes()	77
6.1.2.25 gaiaCreateMetaCatalogTables()	77
6.1.2.26 gaiaCutter()	78
6.1.2.27 gaiaDropTable()	79
6.1.2.28 gaiaDropTable5()	79
6.1.2.29 gaiaDropTableEx()	80
6.1.2.30 gaiaDropTableEx2()	81
6.1.2.31 gaiaDropTableEx3()	81
6.1.2.32 gaiaFreeVectorLayersList()	82
6.1.2.33 gaiaGetLayerExtent()	83
6.1.2.34 gaiaGetVectorLayersList()	83
6.1.2.35 gaiaRenameColumn()	84
6.1.2.36 gaiaRenameTable()	85
6.1.2.37 gaiaStatisticsInvalidate()	86
6.1.2.38 gaiaUpdateMetaCatalogStatistics()	86
6.1.2.39 gaiaUpdateMetaCatalogStatisticsFromMaster()	87
6.1.2.40 insert_epsg_srid()	87
6.1.2.41 is_kml_constant()	88
6.1.2.42 is_tiny_point_enabled()	88
6.1.2.43 load_dbf()	89
6.1.2.44 load_dbf_ex()	90
6.1.2.45 load_dbf_ex2()	90
6.1.2.46 load_dbf_ex3()	91
6.1.2.47 load_geojson()	92
6.1.2.48 load_shapefile()	93
6.1.2.49 load_shapefile_ex()	94
6.1.2.50 load_shapefile_ex2()	95
6.1.2.51 load_shapefile_ex3()	96
6.1.2.52 load_XL()	97
6.1.2.53 load_zip_dbf()	98
6.1.2.54 load_zip_shapefile()	99
6.1.2.55 math_llabs()	100
6.1.2.56 math_round()	100
6.1.2.57 remove_duplicated_rows()	101
6.1.2.58 remove_duplicated_rows_ex()	101
6.1.2.59 remove_duplicated_rows_ex2()	102
6.1.2.60 sanitize_all_geometry_columns()	102

6.1.2.61	sanitize_all_geometry_columns_r()	103
6.1.2.62	sanitize_geometry_column()	104
6.1.2.63	sanitize_geometry_column_r()	105
6.1.2.64	spatial_ref_sys_init()	106
6.1.2.65	spatial_ref_sys_init2()	107
6.1.2.66	spatialite_alloc_connection()	107
6.1.2.67	spatialite_cleanup()	107
6.1.2.68	spatialite_cleanup_ex()	108
6.1.2.69	spatialite_finalize_topologies()	108
6.1.2.70	spatialite_init()	108
6.1.2.71	spatialite_init_ex()	109
6.1.2.72	spatialite_init_geos()	109
6.1.2.73	spatialite_initialize()	110
6.1.2.74	spatialite_set_silent_mode()	110
6.1.2.75	spatialite_set_verbode_mode()	110
6.1.2.76	spatialite_shutdown()	111
6.1.2.77	spatialite_target_cpu()	111
6.1.2.78	spatialite_version()	111
6.1.2.79	srid_get_axis()	111
6.1.2.80	srid_get_datum()	112
6.1.2.81	srid_get_prime_meridian()	112
6.1.2.82	srid_get_projection()	113
6.1.2.83	srid_get_spheroid()	113
6.1.2.84	srid_get_unit()	114
6.1.2.85	srid_has_flipped_axes()	114
6.1.2.86	srid_is_geographic()	115
6.1.2.87	srid_is_projected()	115
6.1.2.88	update_layer_statistics()	116
6.2	src/headers/spatialite/gaiaaux.h File Reference	116
6.2.1	Detailed Description	118
6.2.2	Function Documentation	118
6.2.2.1	gaiaCleanSqlString()	118
6.2.2.2	gaiaConvertCharset()	118
6.2.2.3	gaiaConvertToDMS()	120
6.2.2.4	gaiaConvertToDMSex()	121
6.2.2.5	gaiaConvertToUTF8()	122
6.2.2.6	gaiaCreateMD5Checksum()	123
6.2.2.7	gaiaCreateUTF8Converter()	123
6.2.2.8	gaiaDecodeURL()	124
6.2.2.9	gaiaDequotedSql()	124
6.2.2.10	gaiaDirNameFromPath()	125
6.2.2.11	gaiaDoubleQuotedSql()	125

6.2.2.12	gaiaEncodeURL()	126
6.2.2.13	gaiaFileExtFromPath()	126
6.2.2.14	gaiaFileNameFromPath()	127
6.2.2.15	gaiaFinalizeMD5Checksum()	127
6.2.2.16	gaiaFreeMD5Checksum()	128
6.2.2.17	gaiaFreeUTF8Converter()	128
6.2.2.18	gaiaFullFileNameFromPath()	129
6.2.2.19	gaiaGetLocaleCharset()	129
6.2.2.20	gaiaIllegalSqlName()	129
6.2.2.21	gaiaInsertIntoSqlLog()	130
6.2.2.22	gaiaIsReservedSqliteName()	130
6.2.2.23	gaiaIsReservedSqlName()	131
6.2.2.24	gaiaParseDMS()	131
6.2.2.25	gaiaQuotedSql()	132
6.2.2.26	gaiaRemoveExtraSpaces()	132
6.2.2.27	gaiaSingleQuotedSql()	133
6.2.2.28	gaiaUpdateMD5Checksum()	133
6.2.2.29	gaiaUpdateSqlLog()	134
6.3	src/headers/spatialite/gaiaexif.h File Reference	134
6.3.1	Detailed Description	137
6.3.2	Typedef Documentation	138
6.3.2.1	gaiaExifTagListPtr	138
6.3.2.2	gaiaExifTagPtr	138
6.3.3	Function Documentation	138
6.3.3.1	gaiaExifTagGetByteValue()	138
6.3.3.2	gaiaExifTagGetDoubleValue()	139
6.3.3.3	gaiaExifTagGetFloatValue()	139
6.3.3.4	gaiaExifTagGetHumanReadable()	140
6.3.3.5	gaiaExifTagGetId()	140
6.3.3.6	gaiaExifTagGetLongValue()	141
6.3.3.7	gaiaExifTagGetName()	141
6.3.3.8	gaiaExifTagGetNumValues()	142
6.3.3.9	gaiaExifTagGetRational1Value()	142
6.3.3.10	gaiaExifTagGetRational2Value()	143
6.3.3.11	gaiaExifTagGetRationalValue()	143
6.3.3.12	gaiaExifTagGetShortValue()	144
6.3.3.13	gaiaExifTagGetSignedLongValue()	144
6.3.3.14	gaiaExifTagGetSignedRational1Value()	145
6.3.3.15	gaiaExifTagGetSignedRational2Value()	145
6.3.3.16	gaiaExifTagGetSignedRationalValue()	146
6.3.3.17	gaiaExifTagGetSignedShortValue()	146
6.3.3.18	gaiaExifTagGetStringValue()	147

6.3.3.19	gaiaExifTagGetValueType()	147
6.3.3.20	gaiaExifTagsFree()	147
6.3.3.21	gaiaGetExifGpsTagById()	148
6.3.3.22	gaiaGetExifTagById()	148
6.3.3.23	gaiaGetExifTagByName()	149
6.3.3.24	gaiaGetExifTagByPos()	149
6.3.3.25	gaiaGetExifTags()	150
6.3.3.26	gaiaGetExifTagsCount()	150
6.3.3.27	gaiaGetGpsCoords()	151
6.3.3.28	gaiaGetGpsLatLong()	151
6.3.3.29	gaiaGuessBlobType()	152
6.3.3.30	gaiaIsExifGpsTag()	152
6.4	src/headers/spatialite/gaiageo.h File Reference	153
6.4.1	Detailed Description	153
6.5	src/headers/spatialite/gg_advanced.h File Reference	153
6.5.1	Detailed Description	166
6.5.2	Function Documentation	166
6.5.2.1	gaia3DDistance()	166
6.5.2.2	gaia3dLength()	167
6.5.2.3	gaia3DMaxDistance()	168
6.5.2.4	gaiaAsEncodedPolyLine()	168
6.5.2.5	gaiaAsX3D()	169
6.5.2.6	gaiaAzimuth()	170
6.5.2.7	gaiaBoundary()	171
6.5.2.8	gaiaBoundary_r()	171
6.5.2.9	gaiaConcaveHull()	172
6.5.2.10	gaiaConcaveHull_r()	173
6.5.2.11	gaiaConstrainedDelaunayTriangulation()	174
6.5.2.12	gaiaConstrainedDelaunayTriangulation_r()	174
6.5.2.13	gaiaConvexHull()	175
6.5.2.14	gaiaConvexHull_r()	175
6.5.2.15	gaiaCriticalPointFromGEOSmsg()	176
6.5.2.16	gaiaCriticalPointFromGEOSmsg_r()	176
6.5.2.17	gaiaDegsToRads()	177
6.5.2.18	gaiaDelaunayTriangulation()	178
6.5.2.19	gaiaDelaunayTriangulation_r()	178
6.5.2.20	gaiaDrapeLine()	179
6.5.2.21	gaiaDrapeLineExceptions()	180
6.5.2.22	gaiaEllipsoidAzimuth()	180
6.5.2.23	gaiaFrechetDistance()	181
6.5.2.24	gaiaFrechetDistance_r()	182
6.5.2.25	gaiaFrechetDistanceDensify()	183

6.5.2.26	gaiaFrechetDistanceDensify_r()	183
6.5.2.27	gaiaFromGeos_XY()	184
6.5.2.28	gaiaFromGeos_XY_r()	185
6.5.2.29	gaiaFromGeos_XYM()	185
6.5.2.30	gaiaFromGeos_XYM_r()	186
6.5.2.31	gaiaFromGeos_XYZ()	187
6.5.2.32	gaiaFromGeos_XYZ_r()	187
6.5.2.33	gaiaFromGeos_XYZM()	188
6.5.2.34	gaiaFromGeos_XYZM_r()	189
6.5.2.35	gaiaFromTWKB()	189
6.5.2.36	gaiaGeodesicArea()	190
6.5.2.37	gaiaGeoHash()	191
6.5.2.38	gaiaGeomCollArea()	191
6.5.2.39	gaiaGeomCollArea_r()	192
6.5.2.40	gaiaGeomCollBuffer()	193
6.5.2.41	gaiaGeomCollBuffer_r()	194
6.5.2.42	gaiaGeomCollCentroid()	194
6.5.2.43	gaiaGeomCollCentroid_r()	195
6.5.2.44	gaiaGeomCollContains()	196
6.5.2.45	gaiaGeomCollContains_r()	196
6.5.2.46	gaiaGeomCollCoveredBy()	197
6.5.2.47	gaiaGeomCollCoveredBy_r()	198
6.5.2.48	gaiaGeomCollCovers()	198
6.5.2.49	gaiaGeomCollCovers_r()	199
6.5.2.50	gaiaGeomCollCrosses()	200
6.5.2.51	gaiaGeomCollCrosses_r()	200
6.5.2.52	gaiaGeomCollDisjoint()	201
6.5.2.53	gaiaGeomCollDisjoint_r()	202
6.5.2.54	gaiaGeomCollDistance()	202
6.5.2.55	gaiaGeomCollDistance_r()	203
6.5.2.56	gaiaGeomCollEquals()	204
6.5.2.57	gaiaGeomCollEquals_r()	204
6.5.2.58	gaiaGeomCollIntersects()	205
6.5.2.59	gaiaGeomCollIntersects_r()	206
6.5.2.60	gaiaGeomCollLength()	206
6.5.2.61	gaiaGeomCollLength_r()	207
6.5.2.62	gaiaGeomCollLengthOrPerimeter()	208
6.5.2.63	gaiaGeomCollLengthOrPerimeter_r()	209
6.5.2.64	gaiaGeomCollOverlaps()	209
6.5.2.65	gaiaGeomCollOverlaps_r()	210
6.5.2.66	gaiaGeomCollPreparedContains()	211
6.5.2.67	gaiaGeomCollPreparedCoveredBy()	211

6.5.2.68	gaiaGeomCollPreparedCovers()	212
6.5.2.69	gaiaGeomCollPreparedCrosses()	213
6.5.2.70	gaiaGeomCollPreparedDisjoint()	214
6.5.2.71	gaiaGeomCollPreparedDistance()	215
6.5.2.72	gaiaGeomCollPreparedDistanceWithin()	215
6.5.2.73	gaiaGeomCollPreparedIntersects()	216
6.5.2.74	gaiaGeomCollPreparedOverlaps()	217
6.5.2.75	gaiaGeomCollPreparedTouches()	218
6.5.2.76	gaiaGeomCollPreparedWithin()	219
6.5.2.77	gaiaGeomCollRelate()	219
6.5.2.78	gaiaGeomCollRelate_r()	220
6.5.2.79	gaiaGeomCollRelateBoundaryNodeRule()	221
6.5.2.80	gaiaGeomCollRelateBoundaryNodeRule_r()	222
6.5.2.81	gaiaGeomCollSimplify()	222
6.5.2.82	gaiaGeomCollSimplify_r()	223
6.5.2.83	gaiaGeomCollSimplifyPreserveTopology()	224
6.5.2.84	gaiaGeomCollSimplifyPreserveTopology_r()	224
6.5.2.85	gaiaGeomCollTouches()	225
6.5.2.86	gaiaGeomCollTouches_r()	226
6.5.2.87	gaiaGeomCollWithin()	226
6.5.2.88	gaiaGeomCollWithin_r()	227
6.5.2.89	gaiaGeometryDifference()	228
6.5.2.90	gaiaGeometryDifference_r()	228
6.5.2.91	gaiaGeometryIntersection()	229
6.5.2.92	gaiaGeometryIntersection_r()	230
6.5.2.93	gaiaGeometrySymDifference()	231
6.5.2.94	gaiaGeometrySymDifference_r()	232
6.5.2.95	gaiaGeometryUnion()	233
6.5.2.96	gaiaGeometryUnion_r()	233
6.5.2.97	gaiaGeosConcaveHull()	234
6.5.2.98	gaiaGeosConcaveHull_r()	235
6.5.2.99	gaiaGeosDensify()	236
6.5.2.100	gaiaGeosDensify_r()	236
6.5.2.101	gaiaGeosMakeValid()	237
6.5.2.102	gaiaGeosMakeValid_r()	238
6.5.2.103	gaiaGetGeosAuxErrorMsg()	239
6.5.2.104	gaiaGetGeosAuxErrorMsg_r()	239
6.5.2.105	gaiaGetGeosErrorMsg()	240
6.5.2.106	gaiaGetGeosErrorMsg_r()	240
6.5.2.107	gaiaGetGeosWarningMsg()	241
6.5.2.108	gaiaGetGeosWarningMsg_r()	241
6.5.2.109	gaiaGetPointOnSurface()	242

6.5.2.110	gaiaGetPointOnSurface_r()	243
6.5.2.111	gaiaGetProjDatabasePath()	243
6.5.2.112	gaiaGetProjErrorMsg_r()	244
6.5.2.113	gaiaGetProjString()	245
6.5.2.114	gaiaGetProjWKT()	245
6.5.2.115	gaiaGetRtTopoErrorMsg()	246
6.5.2.116	gaiaGetRtTopoWarningMsg()	246
6.5.2.117	gaiaGuessSridFromWKT()	247
6.5.2.118	gaiaHausdorffDistance()	248
6.5.2.119	gaiaHausdorffDistance_r()	248
6.5.2.120	gaiaHausdorffDistanceDensify()	249
6.5.2.121	gaiaHausdorffDistanceDensify_r()	250
6.5.2.122	gaiaHexagonalGrid()	250
6.5.2.123	gaiaHexagonalGrid_r()	251
6.5.2.124	gaiaHilbertCode()	252
6.5.2.125	gaiaHilbertCode_r()	253
6.5.2.126	gaiaIntersectionMatrixPatternMatch()	253
6.5.2.127	gaiaIntersectionMatrixPatternMatch_r()	254
6.5.2.128	gaialsClosed()	255
6.5.2.129	gaialsClosedGeom()	255
6.5.2.130	gaialsClosedGeom_r()	256
6.5.2.131	gaialsRing()	256
6.5.2.132	gaialsRing_r()	257
6.5.2.133	gaialsSimple()	258
6.5.2.134	gaialsSimple_r()	258
6.5.2.135	gaialsValid()	259
6.5.2.136	gaialsValid_r()	259
6.5.2.137	gaialsValidDetail()	260
6.5.2.138	gaialsValidDetail_r()	261
6.5.2.139	gaialsValidDetailEx()	261
6.5.2.140	gaialsValidDetailEx_r()	262
6.5.2.141	gaialsValidReason()	263
6.5.2.142	gaialsValidReason_r()	263
6.5.2.143	gaiaLargestEmptyCircle()	264
6.5.2.144	gaiaLargestEmptyCircle_r()	265
6.5.2.145	gaiaLineFromEncodedPolyline()	265
6.5.2.146	gaiaLineInterpolateEquidistantPoints()	266
6.5.2.147	gaiaLineInterpolateEquidistantPoints_r()	267
6.5.2.148	gaiaLineInterpolatePoint()	267
6.5.2.149	gaiaLineInterpolatePoint_r()	268
6.5.2.150	gaiaLineLocatePoint()	269
6.5.2.151	gaiaLineLocatePoint_r()	269

6.5.2.152	gaiaLineMerge()	270
6.5.2.153	gaiaLineMerge_r()	271
6.5.2.154	gaiaLinesCutAtNodes()	271
6.5.2.155	gaiaLineSubstring()	272
6.5.2.156	gaiaLineSubstring_r()	273
6.5.2.157	gaiaMakeValid()	274
6.5.2.158	gaiaMakeValidDiscarded()	274
6.5.2.159	gaiaMaxDistance()	275
6.5.2.160	gaiaMaximumInscribedCircle()	276
6.5.2.161	gaiaMaximumInscribedCircle_r()	276
6.5.2.162	gaiaMinimumBoundingCircle()	277
6.5.2.163	gaiaMinimumBoundingCircle_r()	278
6.5.2.164	gaiaMinimumClearance()	279
6.5.2.165	gaiaMinimumClearance_r()	279
6.5.2.166	gaiaMinimumClearanceLine()	280
6.5.2.167	gaiaMinimumClearanceLine_r()	280
6.5.2.168	gaiaMinimumRotatedRectangle()	281
6.5.2.169	gaiaMinimumRotatedRectangle_r()	282
6.5.2.170	gaiaMinimumWidth()	282
6.5.2.171	gaiaMinimumWidth_r()	283
6.5.2.172	gaiaNodeLines()	283
6.5.2.173	gaiaOffsetCurve()	284
6.5.2.174	gaiaOffsetCurve_r()	285
6.5.2.175	gaiaPolygonize()	286
6.5.2.176	gaiaPolygonize_r()	286
6.5.2.177	gaiaProjectedPoint()	287
6.5.2.178	gaiaRadsToDegs()	288
6.5.2.179	gaiaReducePrecision()	288
6.5.2.180	gaiaReducePrecision_r()	289
6.5.2.181	gaiaResetGeosMsg()	290
6.5.2.182	gaiaResetGeosMsg_r()	290
6.5.2.183	gaiaResetProjErrorMsg_r()	291
6.5.2.184	gaiaResetRtTopoMsg()	292
6.5.2.185	gaiaSegmentize()	292
6.5.2.186	gaiaSetGeosAuxErrorMsg()	293
6.5.2.187	gaiaSetGeosAuxErrorMsg_r()	294
6.5.2.188	gaiaSetGeosErrorMsg()	294
6.5.2.189	gaiaSetGeosErrorMsg_r()	295
6.5.2.190	gaiaSetGeosWarningMsg()	295
6.5.2.191	gaiaSetGeosWarningMsg_r()	296
6.5.2.192	gaiaSetProjDatabasePath()	296
6.5.2.193	gaiaSetProjErrorMsg_r()	297

6.5.2.194	gaiaSetRtTopoErrorMsg()	298
6.5.2.195	gaiaSetRtTopoWarningMsg()	298
6.5.2.196	gaiaSharedPaths()	299
6.5.2.197	gaiaSharedPaths_r()	300
6.5.2.198	gaiaShortestLine()	301
6.5.2.199	gaiaShortestLine_r()	301
6.5.2.200	gaiaSingleSidedBuffer()	302
6.5.2.201	gaiaSingleSidedBuffer_r()	303
6.5.2.202	gaiaSnap()	304
6.5.2.203	gaiaSnap_r()	304
6.5.2.204	gaiaSnapToGrid()	305
6.5.2.205	gaiaSplit()	306
6.5.2.206	gaiaSplitLeft()	307
6.5.2.207	gaiaSplitRight()	307
6.5.2.208	gaiaSquareGrid()	308
6.5.2.209	gaiaSquareGrid_r()	309
6.5.2.210	gaiaSubdivide()	310
6.5.2.211	gaiaToGeos()	310
6.5.2.212	gaiaToGeos_r()	311
6.5.2.213	gaiaToGeosSelective()	312
6.5.2.214	gaiaToGeosSelective_r()	312
6.5.2.215	gaiaToTWKB()	313
6.5.2.216	gaiaTransform()	314
6.5.2.217	gaiaTransform_r()	315
6.5.2.218	gaiaTransformEx()	315
6.5.2.219	gaiaTransformEx_r()	316
6.5.2.220	gaiaTransformXY()	317
6.5.2.221	gaiaTransformXY_r()	318
6.5.2.222	gaiaTransformXYZ()	318
6.5.2.223	gaiaTransformXYZ_r()	319
6.5.2.224	gaiaTriangularGrid()	320
6.5.2.225	gaiaTriangularGrid_r()	321
6.5.2.226	gaiaUnaryUnion()	321
6.5.2.227	gaiaUnaryUnion_r()	322
6.5.2.228	gaiaUnionCascaded()	323
6.5.2.229	gaiaUnionCascaded_r()	323
6.5.2.230	gaiaVoronoiDiagram()	324
6.5.2.231	gaiaVoronoiDiagram_r()	325
6.6	src/headers/spatialite/gg_const.h File Reference	326
6.6.1	Detailed Description	333
6.6.2	Macro Definition Documentation	333
6.6.2.1	gaiaGetPoint	333

6.6.2.2	gaiaGetPointXYM	334
6.6.2.3	gaiaGetPointXYZ	335
6.6.2.4	gaiaGetPointXYZM	335
6.6.2.5	gaiaSetPoint	336
6.6.2.6	gaiaSetPointXYM	337
6.6.2.7	gaiaSetPointXYZ	337
6.6.2.8	gaiaSetPointXYZM	338
6.7	src/headers/spatialite/gg_core.h File Reference	338
6.7.1	Detailed Description	345
6.7.2	Function Documentation	345
6.7.2.1	gaiaAddInteriorRing()	345
6.7.2.2	gaiaAddLinestringToGeomColl()	346
6.7.2.3	gaiaAddMeasure()	347
6.7.2.4	gaiaAddPointToGeomColl()	347
6.7.2.5	gaiaAddPointToGeomCollXYM()	348
6.7.2.6	gaiaAddPointToGeomCollXYZ()	348
6.7.2.7	gaiaAddPointToGeomCollXYZM()	348
6.7.2.8	gaiaAddPolygonToGeomColl()	349
6.7.2.9	gaiaAddRingToPolyg()	350
6.7.2.10	gaiaAllocGeomColl()	350
6.7.2.11	gaiaAllocGeomCollXYM()	351
6.7.2.12	gaiaAllocGeomCollXYZ()	351
6.7.2.13	gaiaAllocGeomCollXYZM()	352
6.7.2.14	gaiaAllocLinestring()	352
6.7.2.15	gaiaAllocLinestringXYM()	353
6.7.2.16	gaiaAllocLinestringXYZ()	353
6.7.2.17	gaiaAllocLinestringXYZM()	354
6.7.2.18	gaiaAllocPoint()	354
6.7.2.19	gaiaAllocPointXYM()	355
6.7.2.20	gaiaAllocPointXYZ()	356
6.7.2.21	gaiaAllocPointXYZM()	357
6.7.2.22	gaiaAllocPolygon()	358
6.7.2.23	gaiaAllocPolygonXYM()	358
6.7.2.24	gaiaAllocPolygonXYZ()	359
6.7.2.25	gaiaAllocPolygonXYZM()	359
6.7.2.26	gaiaAllocRing()	360
6.7.2.27	gaiaAllocRingXYM()	361
6.7.2.28	gaiaAllocRingXYZ()	361
6.7.2.29	gaiaAllocRingXYZM()	362
6.7.2.30	gaiaCastGeomCollToXY()	362
6.7.2.31	gaiaCastGeomCollToXYM()	363
6.7.2.32	gaiaCastGeomCollToXYMnoData()	363

6.7.2.33	gaiaCastGeomCollToXYZ()	364
6.7.2.34	gaiaCastGeomCollToXYZM()	365
6.7.2.35	gaiaCastGeomCollToXYZMnoData()	366
6.7.2.36	gaiaCastGeomCollToXYZnoData()	367
6.7.2.37	gaiaCheckClockwise()	367
6.7.2.38	gaiaCheckCounterClockwise()	368
6.7.2.39	gaiaClockwise()	368
6.7.2.40	gaiaCloneGeomColl()	368
6.7.2.41	gaiaCloneGeomCollLinestrings()	369
6.7.2.42	gaiaCloneGeomCollPoints()	369
6.7.2.43	gaiaCloneGeomCollPolygons()	370
6.7.2.44	gaiaCloneGeomCollSpecial()	371
6.7.2.45	gaiaCloneLinestring()	371
6.7.2.46	gaiaCloneLinestringSpecial()	372
6.7.2.47	gaiaClonePolygon()	372
6.7.2.48	gaiaClonePolygonSpecial()	373
6.7.2.49	gaiaCloneRing()	373
6.7.2.50	gaiaCloneRingSpecial()	374
6.7.2.51	gaiaConvertLength()	375
6.7.2.52	gaiaCopyLinestringCoords()	375
6.7.2.53	gaiaCopyLinestringCoordsEx()	376
6.7.2.54	gaiaCopyLinestringCoordsReverse()	376
6.7.2.55	gaiaCopyRingCoords()	377
6.7.2.56	gaiaCopyRingCoordsEx()	377
6.7.2.57	gaiaCopyRingCoordsReverse()	378
6.7.2.58	gaiaCreatePolygon()	378
6.7.2.59	gaiaCurvosityIndex()	379
6.7.2.60	gaiaDimension()	379
6.7.2.61	gaiaDissolvePoints()	380
6.7.2.62	gaiaDissolveSegments()	380
6.7.2.63	gaiaEllipseParams()	381
6.7.2.64	gaiaEnsureClosedRings()	381
6.7.2.65	gaiaExtractLinestringsFromGeomColl()	382
6.7.2.66	gaiaExtractPointsFromGeomColl()	383
6.7.2.67	gaiaExtractPolygonsFromGeomColl()	383
6.7.2.68	gaiaFree()	384
6.7.2.69	gaiaFreeGeomColl()	384
6.7.2.70	gaiaFreeLinestring()	385
6.7.2.71	gaiaFreePoint()	385
6.7.2.72	gaiaFreePolygon()	386
6.7.2.73	gaiaFreeRing()	386
6.7.2.74	gaiaGeodesicArcLength()	387

6.7.2.75	gaiaGeodesicDistance()	387
6.7.2.76	gaiaGeodesicTotalLength()	388
6.7.2.77	gaiaGeometryAliasType()	389
6.7.2.78	gaiaGeometryType()	389
6.7.2.79	gaiaGreatCircleDistance()	390
6.7.2.80	gaiaGreatCircleTotalLength()	391
6.7.2.81	gaiaInsertInteriorRing()	391
6.7.2.82	gaiaInsertLinestringInGeomColl()	392
6.7.2.83	gaiaInsertPolygonInGeomColl()	392
6.7.2.84	gaiaInterpolatePoint()	393
6.7.2.85	gaiaIntersect()	393
6.7.2.86	gaialsEmpty()	394
6.7.2.87	gaialsNotClosedGeomColl()	394
6.7.2.88	gaialsNotClosedGeomColl_r()	395
6.7.2.89	gaialsNotClosedRing()	396
6.7.2.90	gaialsNotClosedRing_r()	397
6.7.2.91	gaialsPointOnPolygonSurface()	398
6.7.2.92	gaialsPointOnRingSurface()	398
6.7.2.93	gaialsToxic()	398
6.7.2.94	gaialsToxic_r()	399
6.7.2.95	gaialsValidTrajectory()	400
6.7.2.96	gaiaLinearize()	400
6.7.2.97	gaiaLineGetPoint()	401
6.7.2.98	gaiaLineSetPoint()	402
6.7.2.99	gaiaLinestringEquals()	402
6.7.2.100	gaiaLocateBetweenMeasures()	403
6.7.2.101	gaiaMakeArc()	404
6.7.2.102	gaiaMakeCircle()	404
6.7.2.103	gaiaMakeEllipse()	405
6.7.2.104	gaiaMakeEllipticArc()	405
6.7.2.105	gaiaMakePolygon()	406
6.7.2.106	gaiaMeasureArea()	406
6.7.2.107	gaiaMeasureLength()	407
6.7.2.108	gaiaMergeGeometries()	408
6.7.2.109	gaiaMergeGeometries_r()	408
6.7.2.110	gaiaMinDistance()	409
6.7.2.111	gaiaNormalizeLonLat()	409
6.7.2.112	gaiaPolygonEquals()	410
6.7.2.113	gaiaReflectCoords()	410
6.7.2.114	gaiaRemoveRepeatedPoints()	411
6.7.2.115	gaiaRingCentroid()	411
6.7.2.116	gaiaRingGetPoint()	412

6.7.2.117	gaiaRingSetPoint()	413
6.7.2.118	gaiaRotateCoords()	414
6.7.2.119	gaiaSanitize()	414
6.7.2.120	gaiaScaleCoords()	415
6.7.2.121	gaiaShiftCoords()	415
6.7.2.122	gaiaShiftCoords3D()	415
6.7.2.123	gaiaShiftLongitude()	416
6.7.2.124	gaiaSwapCoords()	416
6.7.2.125	gaiaTrajectoryInterpolatePoint()	417
6.7.2.126	gaiaUpDownHeight()	418
6.8	src/headers/spatialite/gg_dxf.h File Reference	419
6.8.1	Detailed Description	422
6.8.2	Typedef Documentation	422
6.8.2.1	gaiaDxfArcPtr	422
6.8.2.2	gaiaDxfBlockPtr	423
6.8.2.3	gaiaDxfBoundaryPathPtr	423
6.8.2.4	gaiaDxfCirclePtr	423
6.8.2.5	gaiaDxfExtraAttrPtr	423
6.8.2.6	gaiaDxfHatchPtr	424
6.8.2.7	gaiaDxfHatchSegmPtr	424
6.8.2.8	gaiaDxfHolePtr	424
6.8.2.9	gaiaDxfInsertPtr	424
6.8.2.10	gaiaDxfLayerPtr	425
6.8.2.11	gaiaDxfParserPtr	425
6.8.2.12	gaiaDxfPointPtr	425
6.8.2.13	gaiaDxfPolylinePtr	425
6.8.2.14	gaiaDxfTextPtr	426
6.8.3	Function Documentation	426
6.8.3.1	gaiaCreateDxfParser()	426
6.8.3.2	gaiaDestroyDxfParser()	427
6.8.3.3	gaiaDxfWriteEndSection()	427
6.8.3.4	gaiaDxfWriteEntities()	427
6.8.3.5	gaiaDxfWriteFooter()	428
6.8.3.6	gaiaDxfWriteGeometry()	428
6.8.3.7	gaiaDxfWriteHeader()	429
6.8.3.8	gaiaDxfWriteLayer()	430
6.8.3.9	gaiaDxfWriteLine()	430
6.8.3.10	gaiaDxfWritePoint()	431
6.8.3.11	gaiaDxfWriteRing()	431
6.8.3.12	gaiaDxfWriterInit()	432
6.8.3.13	gaiaDxfWriteTables()	432
6.8.3.14	gaiaDxfWriteText()	433

6.8.3.15	gaiaExportDxf()	433
6.8.3.16	gaiaLoadFromDxfParser()	435
6.8.3.17	gaiaParseDxfFile()	436
6.8.3.18	gaiaParseDxfFile_r()	436
6.9	src/headers/spatialite/gg_dynamic.h File Reference	437
6.9.1	Detailed Description	439
6.9.2	Function Documentation	439
6.9.2.1	gaiaAllocDynamicLine()	439
6.9.2.2	gaiaAppendPointMToDynamicLine()	439
6.9.2.3	gaiaAppendPointToDynamicLine()	440
6.9.2.4	gaiaAppendPointZMToDynamicLine()	440
6.9.2.5	gaiaAppendPointZToDynamicLine()	441
6.9.2.6	gaiaCloneDynamicLine()	441
6.9.2.7	gaiaCreateDynamicLine()	441
6.9.2.8	gaiaDynamicLineDeletePoint()	442
6.9.2.9	gaiaDynamicLineFindByCoords()	442
6.9.2.10	gaiaDynamicLineFindByPos()	443
6.9.2.11	gaiaDynamicLineInsertAfter()	443
6.9.2.12	gaiaDynamicLineInsertBefore()	444
6.9.2.13	gaiaDynamicLineJoinAfter()	445
6.9.2.14	gaiaDynamicLineJoinBefore()	445
6.9.2.15	gaiaDynamicLineSplitAfter()	446
6.9.2.16	gaiaDynamicLineSplitBefore()	446
6.9.2.17	gaiaFreeDynamicLine()	447
6.9.2.18	gaiaPrependPointMToDynamicLine()	447
6.9.2.19	gaiaPrependPointToDynamicLine()	448
6.9.2.20	gaiaPrependPointZMToDynamicLine()	448
6.9.2.21	gaiaPrependPointZToDynamicLine()	449
6.9.2.22	gaiaReverseDynamicLine()	449
6.10	src/headers/spatialite/gg_formats.h File Reference	450
6.10.1	Detailed Description	455
6.10.2	Function Documentation	455
6.10.2.1	gaiaAddDbfField()	456
6.10.2.2	gaiaAllocDbf()	456
6.10.2.3	gaiaAllocDbfField()	457
6.10.2.4	gaiaAllocDbfList()	458
6.10.2.5	gaiaAllocShapefile()	458
6.10.2.6	gaiaAppendToOutBuffer()	458
6.10.2.7	gaiaCloneDbfEntity()	459
6.10.2.8	gaiaCloneDbfField()	459
6.10.2.9	gaiaCloneValue()	460
6.10.2.10	gaiaEndianArch()	461

6.10.2.11	gaiaEwkbGetLinestring()	461
6.10.2.12	gaiaEwkbGetMultiGeometry()	462
6.10.2.13	gaiaEwkbGetPoint()	462
6.10.2.14	gaiaEwkbGetPolygon()	463
6.10.2.15	gaiaExport16()	464
6.10.2.16	gaiaExport32()	464
6.10.2.17	gaiaExport64()	465
6.10.2.18	gaiaExportF32()	465
6.10.2.19	gaiaExportI64()	466
6.10.2.20	gaiaExportU32()	466
6.10.2.21	gaiaFlushDbfHeader()	467
6.10.2.22	gaiaFlushShpHeaders()	467
6.10.2.23	gaiaFreeDbf()	468
6.10.2.24	gaiaFreeDbfField()	468
6.10.2.25	gaiaFreeDbfList()	469
6.10.2.26	gaiaFreeShapefile()	469
6.10.2.27	gaiaFreeValue()	469
6.10.2.28	gaiaFromEWKB()	470
6.10.2.29	gaiaFromFgf()	470
6.10.2.30	gaiaFromSpatialLiteBlobWkb()	471
6.10.2.31	gaiaFromSpatialLiteBlobWkbEx()	472
6.10.2.32	gaiaFromWkb()	472
6.10.2.33	gaiaImport16()	473
6.10.2.34	gaiaImport32()	473
6.10.2.35	gaiaImport64()	474
6.10.2.36	gaiaImportF32()	475
6.10.2.37	gaiaImportI64()	475
6.10.2.38	gaiaImportU32()	476
6.10.2.39	gaiaIsValidDbfList()	476
6.10.2.40	gaiaMakeLine()	477
6.10.2.41	gaiaMakePoint()	477
6.10.2.42	gaiaMakePointEx()	478
6.10.2.43	gaiaMakePointM()	479
6.10.2.44	gaiaMakePointMEx()	479
6.10.2.45	gaiaMakePointZ()	480
6.10.2.46	gaiaMakePointZEx()	480
6.10.2.47	gaiaMakePointZM()	482
6.10.2.48	gaiaMakePointZMEx()	483
6.10.2.49	gaiaMemFseek()	483
6.10.2.50	gaiaMemRead()	484
6.10.2.51	gaiaOpenDbfRead()	484
6.10.2.52	gaiaOpenDbfWrite()	485

6.10.2.53 gaiaOpenDbfWriteEx()	486
6.10.2.54 gaiaOpenShpRead()	486
6.10.2.55 gaiaOpenShpWrite()	487
6.10.2.56 gaiaOpenShpWriteEx()	488
6.10.2.57 gaiaOpenZipDbf()	488
6.10.2.58 gaiaOutBareKml()	489
6.10.2.59 gaiaOutBufferInitialize()	490
6.10.2.60 gaiaOutBufferReset()	490
6.10.2.61 gaiaOutFullKml()	491
6.10.2.62 gaiaOutGeoJSON()	491
6.10.2.63 gaiaOutGml()	492
6.10.2.64 gaiaOutGml_ex()	492
6.10.2.65 gaiaOutLinestringZ()	493
6.10.2.66 gaiaOutLinestringZex()	493
6.10.2.67 gaiaOutPointZ()	494
6.10.2.68 gaiaOutPointZex()	494
6.10.2.69 gaiaOutPolygonZ()	495
6.10.2.70 gaiaOutPolygonZex()	495
6.10.2.71 gaiaOutSvg()	496
6.10.2.72 gaiaOutWkt()	496
6.10.2.73 gaiaOutWktEx()	497
6.10.2.74 gaiaOutWktStrict()	498
6.10.2.75 gaiaParseEWKT()	498
6.10.2.76 gaiaParseGeoJSON()	499
6.10.2.77 gaiaParseGml()	499
6.10.2.78 gaiaParseGml_r()	500
6.10.2.79 gaiaParseHexEWKB()	500
6.10.2.80 gaiaParseKml()	501
6.10.2.81 gaiaParseWkt()	502
6.10.2.82 gaiaReadDbfEntity()	503
6.10.2.83 gaiaReadDbfEntity_ex()	504
6.10.2.84 gaiaReadShpEntity()	504
6.10.2.85 gaiaReadShpEntity_ex()	505
6.10.2.86 gaiaReadWktFromZipShp()	506
6.10.2.87 gaiaResetDbfEntity()	506
6.10.2.88 gaiaSetDoubleValue()	507
6.10.2.89 gaiaSetIntValue()	507
6.10.2.90 gaiaSetNullValue()	508
6.10.2.91 gaiaSetStrValue()	508
6.10.2.92 gaiaShpAnalyze()	508
6.10.2.93 gaiaTextReaderAlloc()	509
6.10.2.94 gaiaTextReaderDestroy()	510

6.10.2.95	gaiaTextReaderFetchField()	510
6.10.2.96	gaiaTextReaderGetRow()	511
6.10.2.97	gaiaTextReaderParse()	511
6.10.2.98	gaiaToCompressedBlobWkb()	512
6.10.2.99	gaiaToEWKB()	512
6.10.2.100	gaiaToEWKT()	513
6.10.2.101	gaiaToFgf()	513
6.10.2.102	gaiaToHexWkb()	514
6.10.2.103	gaiaToSpatialLiteBlobWkb()	514
6.10.2.104	gaiaToSpatialLiteBlobWkbEx()	515
6.10.2.105	gaiaToSpatialLiteBlobWkbEx2()	515
6.10.2.106	gaiaToWkb()	516
6.10.2.107	gaiaWriteDbfEntity()	517
6.10.2.108	gaiaWriteShpEntity()	517
6.10.2.109	gaiaZipfileDbfN()	518
6.10.2.110	gaiaZipfileNumDBF()	518
6.10.2.111	gaiaZipfileNumSHP()	519
6.10.2.112	gaiaZipfileShpN()	519
6.11	src/headers/spatialite/gg_mbr.h File Reference	519
6.11.1	Detailed Description	522
6.11.2	Function Documentation	522
6.11.2.1	gaiaBuildCircleMbr()	522
6.11.2.2	gaiaBuildFilterMbr()	523
6.11.2.3	gaiaBuildMbr()	523
6.11.2.4	gaiaFromSpatialLiteBlobMbr()	524
6.11.2.5	gaiaGetMbrMaxX()	525
6.11.2.6	gaiaGetMbrMaxY()	525
6.11.2.7	gaiaGetMbrMinX()	526
6.11.2.8	gaiaGetMbrMinY()	526
6.11.2.9	gaiaMbrGeometry()	527
6.11.2.10	gaiaMbrLinestring()	527
6.11.2.11	gaiaMbrPolygon()	527
6.11.2.12	gaiaMbrRing()	527
6.11.2.13	gaiaMbrsContains()	528
6.11.2.14	gaiaMbrsDisjoint()	528
6.11.2.15	gaiaMbrsEqual()	529
6.11.2.16	gaiaMbrsIntersects()	529
6.11.2.17	gaiaMbrsOverlaps()	530
6.11.2.18	gaiaMbrsTouches()	530
6.11.2.19	gaiaMbrsWithin()	531
6.11.2.20	gaiaMRangeGeometry()	532
6.11.2.21	gaiaMRangeGeometryEx()	532

6.11.2.22	gaiaMRangeLinestring()	533
6.11.2.23	gaiaMRangeLinestringEx()	533
6.11.2.24	gaiaMRangePolygon()	534
6.11.2.25	gaiaMRangePolygonEx()	534
6.11.2.26	gaiaMRangeRing()	535
6.11.2.27	gaiaMRangeRingEx()	535
6.11.2.28	gaiaParseFilterMbr()	536
6.11.2.29	gaiaZRangeGeometry()	537
6.11.2.30	gaiaZRangeGeometryEx()	537
6.11.2.31	gaiaZRangeLinestring()	538
6.11.2.32	gaiaZRangeLinestringEx()	538
6.11.2.33	gaiaZRangePolygon()	539
6.11.2.34	gaiaZRangePolygonEx()	539
6.11.2.35	gaiaZRangeRing()	540
6.11.2.36	gaiaZRangeRingEx()	540
6.12	src/headers/spatialite/gg_structs.h File Reference	541
6.12.1	Detailed Description	545
6.12.2	Typedef Documentation	545
6.12.2.1	gaiaAttributeFieldDoubleRangePtr	545
6.12.2.2	gaiaAttributeFieldIntRangePtr	545
6.12.2.3	gaiaAttributeFieldMaxSizePtr	546
6.12.2.4	gaiaDbfListPtr	546
6.12.2.5	gaiaDbfPtr	546
6.12.2.6	gaiaDynamicLinePtr	546
6.12.2.7	gaiaGeomCollPtr	547
6.12.2.8	gaiaLayerAttributeFieldPtr	547
6.12.2.9	gaiaLayerAuthPtr	547
6.12.2.10	gaiaLayerExtentPtr	547
6.12.2.11	gaiaLinestringPtr	548
6.12.2.12	gaiaMemFilePtr	548
6.12.2.13	gaiaOutBufferPtr	548
6.12.2.14	gaiaPointPtr	548
6.12.2.15	gaiaPolygonPtr	549
6.12.2.16	gaiaPreRingPtr	549
6.12.2.17	gaiaProjAreaPtr	549
6.12.2.18	gaiaRingPtr	549
6.12.2.19	gaiaShapefilePtr	550
6.12.2.20	gaiaTextReaderPtr	550
6.12.2.21	gaiaVectorLayerPtr	550
6.12.2.22	gaiaVectorLayersListPtr	550
6.13	src/headers/spatialite/gg_wfs.h File Reference	550
6.13.1	Detailed Description	552

6.13.2 Function Documentation	552
6.13.2.1 create_wfs_catalog()	552
6.13.2.2 create_wfs_schema()	553
6.13.2.3 destroy_wfs_catalog()	553
6.13.2.4 destroy_wfs_schema()	554
6.13.2.5 get_wfs_base_describe_url()	554
6.13.2.6 get_wfs_base_request_url()	555
6.13.2.7 get_wfs_catalog_count()	555
6.13.2.8 get_wfs_catalog_item()	555
6.13.2.9 get_wfs_describe_url()	556
6.13.2.10 get_wfs_item_abstract()	556
6.13.2.11 get_wfs_item_name()	557
6.13.2.12 get_wfs_item_title()	557
6.13.2.13 get_wfs_keyword()	558
6.13.2.14 get_wfs_keyword_count()	558
6.13.2.15 get_wfs_layer_srid()	559
6.13.2.16 get_wfs_layer_srid_count()	559
6.13.2.17 get_wfs_request_url()	560
6.13.2.18 get_wfs_schema_column()	560
6.13.2.19 get_wfs_schema_column_count()	561
6.13.2.20 get_wfs_schema_column_info()	561
6.13.2.21 get_wfs_schema_geometry_info()	562
6.13.2.22 get_wfs_version()	563
6.13.2.23 load_from_wfs()	563
6.13.2.24 load_from_wfs_paged()	564
6.13.2.25 load_from_wfs_paged_ex()	565
6.13.2.26 reset_wfs_http_connection()	566
6.14 src/headers/spatialite/gg_xml.h File Reference	566
6.14.1 Detailed Description	569
6.14.2 Function Documentation	569
6.14.2.1 gaia_libxml2_version()	570
6.14.2.2 gaialsCompressedXmlBlob()	570
6.14.2.3 gaialsGpxXmlBlob()	570
6.14.2.4 gaialsIsoMetadataXmlBlob()	571
6.14.2.5 gaialsMapConfigXmlBlob()	571
6.14.2.6 gaialsSchemaValidatedXmlBlob()	572
6.14.2.7 gaialsSldSeRasterStyleXmlBlob()	572
6.14.2.8 gaialsSldSeVectorStyleXmlBlob()	573
6.14.2.9 gaialsSldStyleXmlBlob()	573
6.14.2.10 gaialsSvgXmlBlob()	574
6.14.2.11 gaialsValidXmlBlob()	574
6.14.2.12 gaialsValidXPathExpression()	575

6.14.2.13 gaiaXmlBlobAddFileId()	575
6.14.2.14 gaiaXmlBlobAddParentId()	576
6.14.2.15 gaiaXmlBlobCompression()	577
6.14.2.16 gaiaXmlBlobGetAbstract()	578
6.14.2.17 gaiaXmlBlobGetDocumentSize()	578
6.14.2.18 gaiaXmlBlobGetEncoding()	579
6.14.2.19 gaiaXmlBlobGetFileId()	579
6.14.2.20 gaiaXmlBlobGetGeometry()	580
6.14.2.21 gaiaXmlBlobGetLastParseError()	580
6.14.2.22 gaiaXmlBlobGetLastValidateError()	581
6.14.2.23 gaiaXmlBlobGetLastXPathError()	581
6.14.2.24 gaiaXmlBlobGetName()	582
6.14.2.25 gaiaXmlBlobGetParentId()	583
6.14.2.26 gaiaXmlBlobGetSchemaURI()	583
6.14.2.27 gaiaXmlBlobGetTitle()	584
6.14.2.28 gaiaXmlBlobMLineFromGPX()	584
6.14.2.29 gaiaXmlBlobSetFileId()	585
6.14.2.30 gaiaXmlBlobSetParentId()	586
6.14.2.31 gaiaXmlFromBlob()	586
6.14.2.32 gaiaXmlGetInternalSchemaURI()	587
6.14.2.33 gaiaXmlLoad()	587
6.14.2.34 gaiaXmlStore()	588
6.14.2.35 gaiaXmlTextFromBlob()	589
6.14.2.36 gaiaXmlToBlob()	590
7 Example Documentation	591
7.1 demo1.c	591
7.2 demo2.c	597
7.3 demo3.c	604
7.4 demo4.c	609
7.5 demo5.c	614
Index	621

Chapter 1

Introduction

1.1 Generalities

SpatiaLite is an open source library intended to extend basic SQLite core in order to support full fledged Spatial SQL capabilities.

SQLite is intrinsically simple and lightweight:

- a single lightweight library implementing the full SQL engine.
- standard SQL implementation: almost complete SQL-92.
- no complex client/server architecture.
- a whole database simply corresponds to a single monolithic file [no size limits].
- any DB-file can be safely exchanged across different platforms, because the internal architecture is universally portable.
- no installation, no configuration.

SpatiaLite is smoothly integrated into SQLite so to deploy a complete and powerful Spatial DBMS [mostly OGC-↔SFS compliant].

All this fully preserving the lightness and simplicity typical of SQLite itself.

That's not all: SpatiaLite supports direct SQL access to several commonly used *external datasources*, this including:

- **ESRI Shapefiles**.
- **DBF** Archive Files.
- **TXT/CSV** structured text files.
- **Spreadsheets** [.xls format].

And SpatiaLite actively supports many alternative standard Geometry notations:

- **WKT** [Well Known Text] and **WKB** [Well Known Binary].

- PostGIS own **EWKT** and **EWKB** [Extended WKT / WKB].
- **GML** [Geography Markup Language, both v2 and v3].
- **KML** [Keyhole Markup Language, used by Google Maps and Google Earth].
- **GeoJSON** [Geometry Java Script Object Notation].
- **SVG** [Scalable Vector Graphics].

Conclusion: using SQLite + Spatialite you can deploy an alternative Spatial DBMS roughly equivalent to PostgreSQL + PostGIS.

The main difference between them isn't in powerness, but mainly relies on architecture:

- PostgreSQL + PostGIS fully supports a client/server architecture.
This is well fit for complex and sophisticated Spatial Data infrastructures, but surely implies a certain degree of complexity.
- SQLite + Spatialite supports a much more simplest personal architecture.
This is most appropriate for desktop, stand-alone, personal activities.

Choosing the one or the other simply depends on your very specific requirements:

- no one is better than the other one: they are simply optimized for different environments.
- both them can roughly support the same Spatial Data processing capabilities.
- feel free to choose the best fit one accordingly to your effective goals.

1.2 Building

Building and installing the Spatialite library is straightforward:

```
./configure
make
make install
```

Please note: Spatialite depends on the following open source libraries:

- **GNU ICONV**
locale charset encodings support
- **GEOS**
Geometry engine
- **PROJ.4**
Spatial Reference System handling [coordinate re-projection]
- **FreeXL**
Spreadsheet input support [.xls format]

The library comes in two different flavors:

- **libspatialite**

standard, canonical library: the best and safest way to deploy Spatialite.

this obviously depends on *external* **libsqlite**: thus ensuring full coherence between libraries.

warmly recommended, mostly on Unix-like systems.

- **libspatialite-amalgamation**

The whole library is *amalgamated* into a single monolithic file and includes an *internal private copy* of **libsqlite**.

Using the *amalgamated* library may strongly simplify any following installation process, and nicely supports **static linkage**.

Anyway, you can safely apply the *amalgamated* approach only to self-standing apps.

Attempting to use the *amalgated* library on complex frameworks or on data connectors / language bindings may easily cause serious conflicts.

1.3 Deployment

You can deploy Spatialite in two alternative ways:

- you can load the Spatialite library as a **dynamic extension** to SQLite.
This allows SQLite to support SQL Spatial Data [Geometry] and SQL Spatial Functions.
Theoretically, any generic tool or language connector supporting SQLite can support this *extension* mechanism;
sadly enough, sometimes this feature is intentionally disabled: I'm sorry for you if this is your specific case.

How to load Spatialite as a dynamic extension to SQLite:

```
SELECT load_extension('spatialite_dynamic_library_name');
```

- you can directly link the Spatialite library to any application of your own.
This allows you to ship a complete, powerful, self-contained Spatial SQL engine directly supporting your app.
And such Spatial SQL engine doesn't requires any installation or configuration at all.
That's not all: linking the Spatialite to your own C/C++ code you aren't simply constrained to use SQL: adopting this approach you can directly access the complete C API.

Linking Spatialite to your own code is usually simple:

```
gcc my_program.c -o my_program -lspatialite
```

On some systems you may have to provide a slightly more complex arrangement:

```
gcc -I/usr/local/include my_program.c -o my_program \
  -L/usr/local/lib -lspatialite -lsqlite -lgeos_c -lgeos \
  -lproj -lfreeexl -liconv -lm -lstdc++
```

Spatialite also provides pkg-config support, so you can also do

```
gcc -I/usr/local/include my_program.c -o my_program \
  `pkg-config --libs spatialite`
```

1.4 License

Spatialite is licensed under the MPL tri-license terms: you are free to choose the best-fit license between:

- the MPL 1.1
- the GPL v2.0 or any subsequent version
- the LGPL v2.1 or any subsequent version

Enjoy, and happy coding

Chapter 2

Deprecated List

Global [gaiaDropTableEx3](#) (sqlite3 *sqlite, const char *prefix, const char *table, int transaction, char **error_message)
use [gaiaDropTable5\(\)](#) as a full replacement

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

gaia_dxf_arc	
Wrapper for DXF Arc object	11
gaia_dxf_block	
Wrapper for DXF Block object	12
gaia_dxf_boundary_path	
Wrapper for DXF Boundary Path object	13
gaia_dxf_circle	
Wrapper for DXF Circle object	14
gaia_dxf_extra_attr	
Wrapper for DXF Extra Attribute object	15
gaia_dxf_hatch	
Wrapper for DXF Pattern Hatch object	15
gaia_dxf_hatch_segm	
Wrapper for DXF Pattern Segment object	16
gaia_dxf_hole	
Wrapper for DXF Polygon interior hole object	17
gaia_dxf_insert	
Wrapper for DXF Insert object	18
gaia_dxf_layer	
Wrapper for DXF Layer object	19
gaia_dxf_parser	
Wrapper for DXF Parser object	22
gaia_dxf_point	
Wrapper for DXF Point object	25
gaia_dxf_polyline	
Wrapper for DXF Polyline object could be a Linestring or a Polygon depending on the is_closed flag	26
gaia_dxf_text	
Wrapper for DXF Text object	27
gaia_dxf_write	
Wrapper for DXF Write object	28
gaiaAttributeFieldDoubleRangeInfos	
Attribute/Field Double range infos	28
gaiaAttributeFieldIntRangeInfos	
Attribute/Field Integer range infos	29

gaiaAttributeFieldMaxSizeInfos	
Attribute/Field MaxSize/Length infos	29
gaiaDbfFieldStruct	
Container for DBF field	30
gaiaDbfListStruct	
Container for a list of DBF fields	31
gaiaDbfStruct	
Container for DBF file handling	32
gaiaDynamicLineStruct	
Container for dynamically growing line/ring	33
gaiaExifTagListStruct	
Container for a list of EXIF tags	34
gaiaExifTagStruct	
Container for an EXIF tag	35
gaiaGeomCollStruct	
Container for OGC GEOMETRYCOLLECTION Geometry	36
gaiaLayerAttributeFieldInfos	
LayerAttributeField infos	37
gaiaLayerAuthInfos	
Layer Auth infos	38
gaiaLayerExtentInfos	
Layer Extent infos	39
gaiaLinestringStruct	
Container for OGC LINESTRING Geometry	40
gaiaMemFileStruct	
A Memory based File	41
gaiaOutBufferStruct	
Container for dynamically growing output buffer	41
gaiaPointStruct	
Container for OGC POINT Geometry	42
gaiaPolygonStruct	
Container for OGC POLYGON Geometry	43
gaiaPreRingStruct	
Container similar to LINESTRING [internally used]	44
gaiaProjAreaStr	
BBOX corresponding to PROJ.6 AREA	45
gaiaRingStruct	
Container for OGC RING Geometry	45
gaiaShapefileStruct	
Container for SHP file handling	46
gaiaValueStruct	
Container for variant (multi-type) value	48
gaiaVectorLayerItem	
Vector Layer item	49
gaiaVectorLayersListStr	
Container for Vector Layers List	50
vrttxt_column_header	
Container for Virtual Text column (field) header	51
vrttxt_line	
Container for Virtual Text record (line)	51
vrttxt_reader	
Container for Virtual Text file handling	52
vrttxt_row	
Container for Virtual Text record (line) offsets	53
vrttxt_row_block	
Container for Virtual Text block of records	54

Chapter 4

File Index

4.1 File List

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

src/headers/ spatialite.h	
Main Spatialite header file	55
src/headers/spatialite/ gaiaaux.h	
Auxiliary/helper functions	116
src/headers/spatialite/ gaiaexif.h	
EXIF/image: supporting functions and constants	134
src/headers/spatialite/ gaiageo.h	
Geometry handling functions and constants	153
src/headers/spatialite/ gg_advanced.h	
Geometry handling functions: advanced	153
src/headers/spatialite/ gg_const.h	
Geometry constants and macros	326
src/headers/spatialite/ gg_core.h	
Geometry handling functions: core	338
src/headers/spatialite/ gg_dxf.h	
Geometry handling functions: DXF files	419
src/headers/spatialite/ gg_dynamic.h	
Geometry handling functions: DynamicLine handling	437
src/headers/spatialite/ gg_formats.h	
Geometry handling functions: formats	450
src/headers/spatialite/ gg_mbr.h	
Geometry handling functions: MBR	519
src/headers/spatialite/ gg_structs.h	
Geometry structures	541
src/headers/spatialite/ gg_wfs.h	
WFS support	550
src/headers/spatialite/ gg_xml.h	
Geometry handling functions: XML document	566

Chapter 5

Data Structure Documentation

5.1 gaia_dxf_arc Struct Reference

wrapper for DXF Arc object

```
#include <gg_dxf.h>
```

Data Fields

- double [cx](#)
Center X coordinate.
- double [cy](#)
Center Y coordinate.
- double [cz](#)
Center Z coordinate.
- double [radius](#)
radius
- double [start](#)
start angle
- double [stop](#)
stop angle

5.1.1 Detailed Description

wrapper for DXF Arc object

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

- [src/headers/spatialite/gg_dxf.h](#)

- pointer to first DXF Hatch object [linked list]*
- [gaiaDxfHatchPtr last_hatch](#)

pointer to last DXF Hatch object [linked list]
- int [is3Dtext](#)

boolean flag: contains 3d Text objects
- int [is3Dpoint](#)

boolean flag: contains 3d Point objects
- int [is3Dline](#)

boolean flag: contains 3d Polyline (Linestring) objects
- int [is3Dpolyg](#)

boolean flag: contains 3d Polyline (Polygon) objects
- struct [gaia_dxf_block](#) * [next](#)

pointer to next item [linked list]

5.2.1 Detailed Description

wrapper for DXF Block object

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

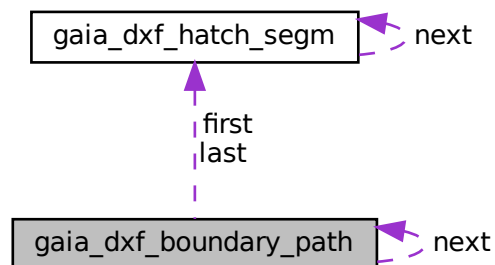
- [src/headers/spatialite/gg_dxf.h](#)

5.3 gaia_dxf_boundary_path Struct Reference

wrapper for DXF Boundary Path object

```
#include <gg_dxf.h>
```

Collaboration diagram for `gaia_dxf_boundary_path`:



Data Fields

- [gaiaDxfHatchSegmPtr first](#)
pointer to first segment
- [gaiaDxfHatchSegmPtr last](#)
pointer to last segment
- struct [gaia_dxf_boundary_path](#) * [next](#)
pointer to next item [linked list]

5.3.1 Detailed Description

wrapper for DXF Boundary Path object

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

- [src/headers/spatialite/gg_dxf.h](#)

5.4 [gaia_dxf_circle](#) Struct Reference

wrapper for DXF Circle object

```
#include <gg_dxf.h>
```

Data Fields

- double [cx](#)
Center X coordinate.
- double [cy](#)
Center Y coordinate.
- double [cz](#)
Center Z coordinate.
- double [radius](#)
radius

5.4.1 Detailed Description

wrapper for DXF Circle object

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

- [src/headers/spatialite/gg_dxf.h](#)

Data Fields

- double [spacing](#)
hatch pattern spacing
- double [angle](#)
hatch line angle
- double [base_x](#)
hatch line base X
- double [base_y](#)
hatch line base Y
- double [offset_x](#)
hatch line offset X
- double [offset_y](#)
hatch line offset Y
- [gaiaDxfBoundaryPathPtr](#) first
pointer to first Boundary
- [gaiaDxfBoundaryPathPtr](#) last
pointer to last Boundary
- [gaiaGeomCollPtr](#) boundary
pointer to Boundary geometry
- [gaiaDxfHatchSegmPtr](#) first_out
pointer to first Pattern segment
- [gaiaDxfHatchSegmPtr](#) last_out
pointer to last Pattern segment
- struct [gaia_dxf_hatch](#) * next
pointer to next item [linked list]

5.6.1 Detailed Description

wrapper for DXF Pattern Hatch object

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

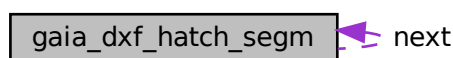
- [src/headers/spatialite/gg_dxf.h](#)

5.7 [gaia_dxf_hatch_seg](#) Struct Reference

wrapper for DXF Pattern Segment object

```
#include <gg_dxf.h>
```

Collaboration diagram for [gaia_dxf_hatch_seg](#):



Data Fields

- double [x0](#)
start X
- double [y0](#)
start Y
- double [x1](#)
end X
- double [y1](#)
end Y
- struct [gaia_dxf_hatch_segm](#) * [next](#)
pointer to next item [linked list]

5.7.1 Detailed Description

wrapper for DXF Pattern Segment object

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

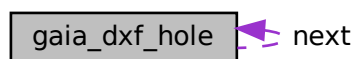
- [src/headers/spatialite/gg_dxf.h](#)

5.8 gaia_dxf_hole Struct Reference

wrapper for DXF Polygon interior hole object

```
#include <gg_dxf.h>
```

Collaboration diagram for `gaia_dxf_hole`:



Data Fields

- int [points](#)
total count of points
- double * [x](#)
array of X coordinates
- double * [y](#)
array of Y coordinates
- double * [z](#)
array of Z coordinates
- struct [gaia_dxf_hole](#) * [next](#)
pointer to next item [linked list]

5.8.1 Detailed Description

wrapper for DXF Polygon interior hole object

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

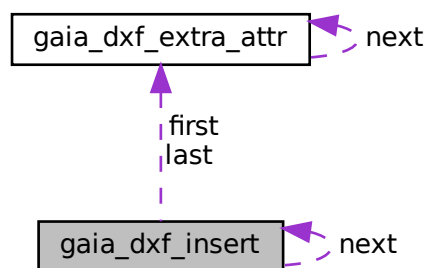
- [src/headers/spatialite/gg_dxf.h](#)

5.9 gaia_dxf_insert Struct Reference

wrapper for DXF Insert object

```
#include <gg_dxf.h>
```

Collaboration diagram for gaia_dxf_insert:



Data Fields

- `char * block_id`
pointer to Block ID string
- `double x`
X coordinate.
- `double y`
Y coordinate.
- `double z`
Z coordinate.
- `double scale_x`
X scale factor.
- `double scale_y`
Y scale factor.
- `double scale_z`
Z scale factor.
- `double angle`
rotation angle

- int [hasText](#)
boolean flag: contains Text objects
- int [hasPoint](#)
boolean flag: contains Point objects
- int [hasLine](#)
boolean flag: contains Polyline (Linestring) objects
- int [hasPolyg](#)
boolean flag: contains Polyline (Polygon) objects
- int [hasHatch](#)
boolean flag: contains Hatch objects
- int [is3Dtext](#)
boolean flag: contains 3d Text objects
- int [is3Dpoint](#)
boolean flag: contains 3d Point objects
- int [is3Dline](#)
boolean flag: contains 3d Polyline (Linestring) objects
- int [is3Dpolyg](#)
boolean flag: contains 3d Polyline (Polygon) objects
- [gaiaDxfExtraAttrPtr](#) first
pointer to first Extra Attribute [linked list]
- [gaiaDxfExtraAttrPtr](#) last
pointer to last Extra Attribute [linked list]
- struct [gaia_dxf_insert](#) * next
pointer to next item [linked list]

5.9.1 Detailed Description

wrapper for DXF Insert object

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

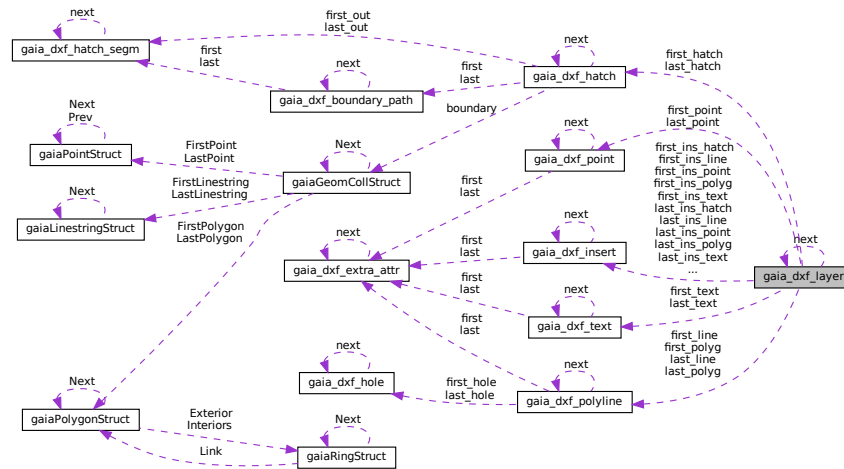
- [src/headers/spatialite/gg_dxf.h](#)

5.10 gaia_dxf_layer Struct Reference

wrapper for DXF Layer object

```
#include <gg_dxf.h>
```

Collaboration diagram for `gaia_dxf_layer`:



Data Fields

- `char * layer_name`
pointer to Layer Name string
- `gaiaDxfTextPtr first_text`
pointer to first DXF Text object [linked list]
- `gaiaDxfTextPtr last_text`
pointer to last DXF Text object [linked list]
- `gaiaDxfPointPtr first_point`
pointer to first DXF Point object [linked list]
- `gaiaDxfPointPtr last_point`
pointer to last DXF Point object [linked list]
- `gaiaDxfPolylinePtr first_line`
pointer to first DXF Polyline (Linestring) object [linked list]
- `gaiaDxfPolylinePtr last_line`
pointer to last DXF Polyline (Linestring) object [linked list]
- `gaiaDxfPolylinePtr first_polyg`
pointer to first DXF Polyline (Polygon) object [linked list]
- `gaiaDxfPolylinePtr last_polyg`
pointer to last DXF Polyline (Polygon) object [linked list]
- `gaiaDxfHatchPtr first_hatch`
pointer to first DXF Hatch object [linked list]
- `gaiaDxfHatchPtr last_hatch`
pointer to last DXF Hatch object [linked list]
- `gaiaDxfInsertPtr first_ins_text`
pointer to first DXF Insert Text object [linked list]
- `gaiaDxfInsertPtr last_ins_text`
pointer to last DXF Insert Text object [linked list]
- `gaiaDxfInsertPtr first_ins_point`
pointer to first DXF Insert Point object [linked list]
- `gaiaDxfInsertPtr last_ins_point`
pointer to last DXF Insert Point object [linked list]

- pointer to last DXF Insert Point object [linked list]*
- [gaiaDxfInsertPtr first_ins_line](#)
 - pointer to first DXF Insert Polyline (Linestring) object [linked list]*
- [gaiaDxfInsertPtr last_ins_line](#)
 - pointer to last DXF Insert Polyline (Linestring) object [linked list]*
- [gaiaDxfInsertPtr first_ins_polyg](#)
 - pointer to first DXF Insert Polyline (Polygon) object [linked list]*
- [gaiaDxfInsertPtr last_ins_polyg](#)
 - pointer to last DXF Insert Polyline (Polygon) object [linked list]*
- [gaiaDxfInsertPtr first_ins_hatch](#)
 - pointer to first DXF Insert Hatch object [linked list]*
- [gaiaDxfInsertPtr last_ins_hatch](#)
 - pointer to last DXF Insert Hatch object [linked list]*
- int [is3Dtext](#)
 - boolean flag: contains 3d Text objects*
- int [is3Dpoint](#)
 - boolean flag: contains 3d Point objects*
- int [is3Dline](#)
 - boolean flag: contains 3d Polyline (Linestring) objects*
- int [is3Dpolyg](#)
 - boolean flag: contains 3d Polyline (Polygon) objects*
- int [is3DinsText](#)
 - boolean flag: contains 3d Insert Text objects*
- int [is3DinsPoint](#)
 - boolean flag: contains 3d Insert Point objects*
- int [is3DinsLine](#)
 - boolean flag: contains 3d Insert Polyline (Linestring) objects*
- int [is3DinsPolyg](#)
 - boolean flag: contains 3d Insert Polyline (Polygon) objects*
- int [hasExtraText](#)
 - boolean flag: contains Text Extra Attributes*
- int [hasExtraPoint](#)
 - boolean flag: contains Point Extra Attributes*
- int [hasExtraLine](#)
 - boolean flag: contains Polyline (Linestring) Extra Attributes*
- int [hasExtraPolyg](#)
 - boolean flag: contains Polyline (Polygon) Extra Attributes*
- int [hasExtraInsText](#)
 - boolean flag: contains Insert Text Extra Attributes*
- int [hasExtraInsPoint](#)
 - boolean flag: contains Insert Text Extra Attributes*
- int [hasExtraInsLine](#)
 - boolean flag: contains Insert Polyline (Linestring) Extra Attributes*
- int [hasExtraInsPolyg](#)
 - boolean flag: contains Insert Polyline (Polygon) Extra Attributes*
- struct [gaia_dxf_layer](#) * [next](#)
 - pointer to next item [linked list]*

- OUT: pointer to last DXF Block object [linked list].*
- int [force_dims](#)

IN: parser option - dimension handling.
 - int [srid](#)

IN: parser option - the SRID.
 - const char * [selected_layer](#)

IN: parser option - pointer the single Layer Name string.
 - const char * [prefix](#)

IN: parser option - pointer to prefix string for DB tables.
 - int [linked_rings](#)

IN: parser option - linked rings special handling.
 - int [unlinked_rings](#)

IN: parser option - unlinked rings special handling.
 - int [line_no](#)

internal parser variable
 - int [op_code_line](#)

internal parser variable
 - int [op_code](#)

internal parser variable
 - int [section](#)

internal parser variable
 - int [tables](#)

internal parser variable
 - int [blocks](#)

internal parser variable
 - int [entities](#)

internal parser variable
 - int [is_layer](#)

internal parser variable
 - int [is_block](#)

internal parser variable
 - int [is_text](#)

internal parser variable
 - int [is_point](#)

internal parser variable
 - int [is_polyline](#)

internal parser variable
 - int [is_lwpolyline](#)

internal parser variable
 - int [is_line](#)

internal parser variable
 - int [is_circle](#)

internal parser variable
 - int [is_arc](#)

internal parser variable
 - int [is_vertex](#)

internal parser variable
 - int [is_hatch](#)

internal parser variable
 - int [is_hatch_boundary](#)

internal parser variable

- int [is_insert](#)
internal parser variable
- int [eof](#)
internal parser variable
- int [error](#)
internal parser variable
- char * [curr_layer_name](#)
internal parser variable
- [gaiaDxfText](#) [curr_text](#)
internal parser variable
- [gaiaDxfInsert](#) [curr_insert](#)
internal parser variable
- [gaiaDxfBlock](#) [curr_block](#)
internal parser variable
- [gaiaDxfPoint](#) [curr_point](#)
internal parser variable
- [gaiaDxfPoint](#) [curr_end_point](#)
internal parser variable
- [gaiaDxfCircle](#) [curr_circle](#)
internal parser variable
- [gaiaDxfArc](#) [curr_arc](#)
internal parser variable
- int [is_closed_polyline](#)
internal parser variable
- [gaiaDxfPointPtr](#) [first_pt](#)
internal parser variable
- [gaiaDxfPointPtr](#) [last_pt](#)
internal parser variable
- char * [extra_key](#)
internal parser variable
- char * [extra_value](#)
internal parser variable
- [gaiaDxfExtraAttrPtr](#) [first_ext](#)
internal parser variable
- [gaiaDxfExtraAttrPtr](#) [last_ext](#)
internal parser variable
- [gaiaDxfHatchPtr](#) [curr_hatch](#)
internal parser variable
- int [undeclared_layers](#)
internal parser variable

5.11.1 Detailed Description

wrapper for DXF Parser object

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

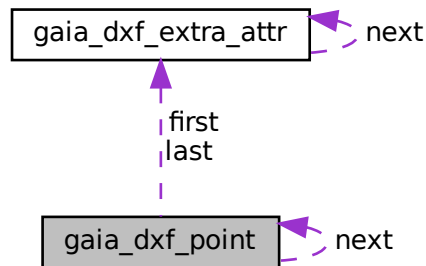
- [src/headers/spatialite/gg_dxf.h](#)

5.12 gaia_dxf_point Struct Reference

wrapper for DXF Point object

```
#include <gg_dxf.h>
```

Collaboration diagram for gaia_dxf_point:



Data Fields

- double [x](#)
X coordinate.
- double [y](#)
Y coordinate.
- double [z](#)
Z coordinate.
- [gaiaDxfExtraAttrPtr](#) [first](#)
pointer to first Extra Attribute [linked list]
- [gaiaDxfExtraAttrPtr](#) [last](#)
pointer to last Extra Attribute [linked list]
- struct [gaia_dxf_point](#) * [next](#)
pointer to next item [linked list]

5.12.1 Detailed Description

wrapper for DXF Point object

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

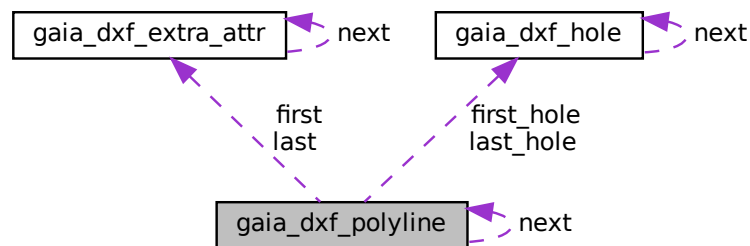
- [src/headers/spatialite/gg_dxf.h](#)

5.13 gaia_dxf_polyline Struct Reference

wrapper for DXF Polyline object could be a Linestring or a Polygon depending on the is_closed flag

```
#include <gg_dxf.h>
```

Collaboration diagram for gaia_dxf_polyline:



Data Fields

- int `is_closed`
open (Linestring) or closed (Polygon exterior ring)
- int `points`
total count of points
- double * `x`
array of X coordinates
- double * `y`
array of Y coordinates
- double * `z`
array of Z coordinates
- `gaiaDxfHolePtr` `first_hole`
pointer to first Polygon hole [linked list]
- `gaiaDxfHolePtr` `last_hole`
pointer to last Polygon hole [linked list]
- `gaiaDxfExtraAttrPtr` `first`
pointer to first Extra Attribute [linked list]
- `gaiaDxfExtraAttrPtr` `last`
pointer to last Extra Attribute [linked list]
- struct `gaia_dxf_polyline` * `next`
pointer to next item [linked list]

5.13.1 Detailed Description

wrapper for DXF Polyline object could be a Linestring or a Polygon depending on the is_closed flag

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

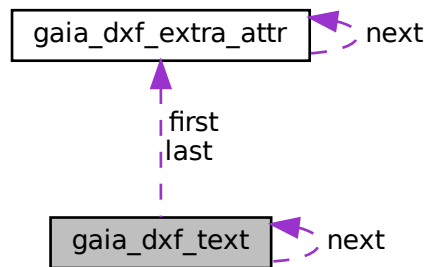
- `src/headers/spatialite/gg_dxf.h`

5.14 gaia_dxf_text Struct Reference

wrapper for DXF Text object

```
#include <gg_dxf.h>
```

Collaboration diagram for gaia_dxf_text:



Data Fields

- char * [label](#)
pointer to Label string
- double [x](#)
X coordinate.
- double [y](#)
Y coordinate.
- double [z](#)
Z coordinate.
- double [angle](#)
label rotation angle
- [gaiaDxfExtraAttrPtr](#) [first](#)
pointer to first Extra Attribute [linked list]
- [gaiaDxfExtraAttrPtr](#) [last](#)
pointer to last Extra Attribute [linked list]
- struct [gaia_dxf_text](#) * [next](#)
pointer to next item [linked list]

5.14.1 Detailed Description

wrapper for DXF Text object

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

- [src/headers/spatialite/gg_dxf.h](#)

5.15 gaia_dxf_write Struct Reference

wrapper for DXF Write object

```
#include <gg_dxf.h>
```

Data Fields

- FILE * [out](#)
IN: output DXF file handle.
- int [precision](#)
IN: coord's precision (number of decimal digits)
- int [version](#)
IN: DXF version number.
- int [count](#)
OUT: count of exported geometries.
- int [error](#)
OUT: error flag.

5.15.1 Detailed Description

wrapper for DXF Write object

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

- [src/headers/spatialite/gg_dxf.h](#)

5.16 gaiaAttributeFieldDoubleRangeInfos Struct Reference

Attribute/Field Double range infos.

```
#include <gg_structs.h>
```

Data Fields

- double [MinValue](#)
Minimum value.
- double [MaxValue](#)
Maximum value.

5.16.1 Detailed Description

Attribute/Field Double range infos.

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

- [src/headers/spatialite/gg_structs.h](#)

5.17 gaiaAttributeFieldIntRangeInfos Struct Reference

Attribute/Field Integer range infos.

```
#include <gg_structs.h>
```

Data Fields

- sqlite3_int64 [MinValue](#)
Minimum value.
- sqlite3_int64 [MaxValue](#)
Maximum value.

5.17.1 Detailed Description

Attribute/Field Integer range infos.

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

- src/headers/spatialite/[gg_structs.h](#)

5.18 gaiaAttributeFieldMaxSizeInfos Struct Reference

Attribute/Field MaxSize/Length infos.

```
#include <gg_structs.h>
```

Data Fields

- int [MaxSize](#)
MaxSize / MaxLength.

5.18.1 Detailed Description

Attribute/Field MaxSize/Length infos.

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

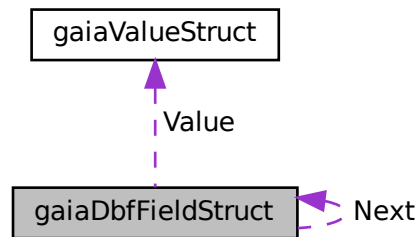
- src/headers/spatialite/[gg_structs.h](#)

5.19 gaiaDbfFieldStruct Struct Reference

Container for DBF field.

```
#include <gg_structs.h>
```

Collaboration diagram for gaiaDbfFieldStruct:



Data Fields

- char * [Name](#)
field name
- unsigned char [Type](#)
DBF data type.
- int [Offset](#)
DBF buffer offset [where the field value starts].
- unsigned char [Length](#)
total DBF buffer field length (in bytes)
- unsigned char [Decimals](#)
precision (decimal digits)
- [gaiaValuePtr](#) [Value](#)
current variant [multi-type] value
- struct [gaiaDbfFieldStruct](#) * [Next](#)
pointer to next item [linked list]

5.19.1 Detailed Description

Container for DBF field.

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

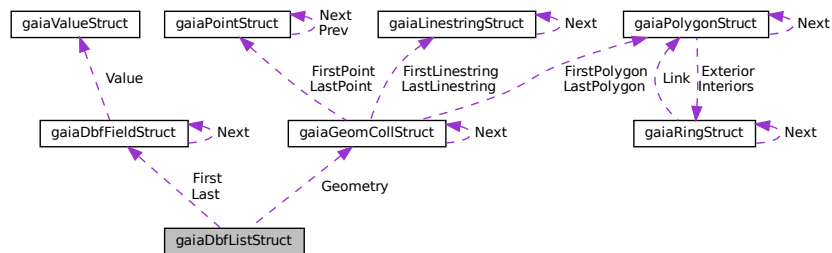
- [src/headers/spatialite/gg_structs.h](#)

5.20 gaiaDbfListStruct Struct Reference

Container for a list of DBF fields.

```
#include <gg_structs.h>
```

Collaboration diagram for gaiaDbfListStruct:



Data Fields

- `int RowId`
current RowID
- `gaiaGeomCollPtr Geometry`
current Geometry
- `gaiaDbfFieldPtr First`
pointer to first DBF field [linked list]
- `gaiaDbfFieldPtr Last`
pointer to last DBF field [linked list]

5.20.1 Detailed Description

Container for a list of DBF fields.

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

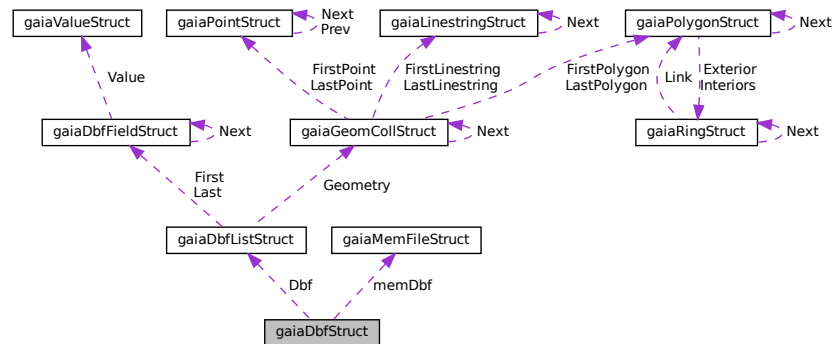
- `src/headers/spatialite/gg_structs.h`

5.21 gaiaDbfStruct Struct Reference

Container for DBF file handling.

```
#include <gg_structs.h>
```

Collaboration diagram for gaiaDbfStruct:



Data Fields

- int [endian_arch](#)
DBF endian arch.
- int [Valid](#)
validity flag: 1 = ready to be processed
- char * [Path](#)
DBF file pathname.
- FILE * [fIDbf](#)
FILE handle.
- [gaiaMemFilePtr](#) [memDbf](#)
Memory based DBF file.
- [gaiaDbfListPtr](#) [Dbf](#)
list of DBF fields
- unsigned char * [BufDbf](#)
I/O buffer.
- int [DbfHdsz](#)
header size (in bytes)
- int [DbfReclen](#)
record length (in bytes)
- int [DbfSize](#)
current file size
- int [DbfRecno](#)
current Record Number
- void * [IconvObj](#)
handle to ICONV converter object
- char * [LastError](#)
last error message (may be NULL)

5.21.1 Detailed Description

Container for DBF file handling.

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

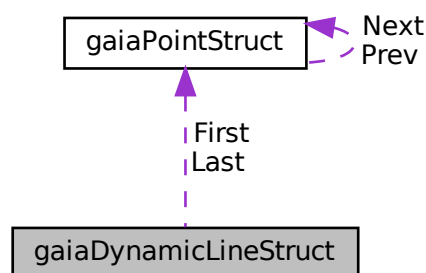
- [src/headers/spatialite/gg_structs.h](#)

5.22 gaiaDynamicLineStruct Struct Reference

Container for dynamically growing line/ring.

```
#include <gg_structs.h>
```

Collaboration diagram for gaiaDynamicLineStruct:



Data Fields

- `int` [Error](#)
invalid object
- `int` [Srid](#)
the SRID
- `gaiaPointPtr` [First](#)
pointer to first POINT [double linked list]
- `gaiaPointPtr` [Last](#)
pointer to last POINT [double linked list]

5.22.1 Detailed Description

Container for dynamically growing line/ring.

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

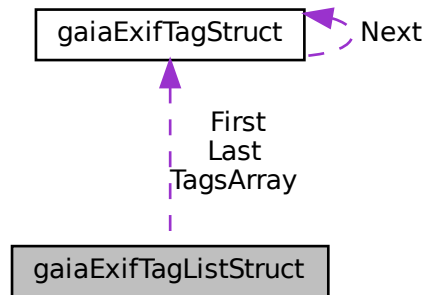
- [src/headers/spatialite/gg_structs.h](#)

5.23 gaiaExifTagListStruct Struct Reference

Container for a list of EXIF tags.

```
#include <gaiaexif.h>
```

Collaboration diagram for gaiaExifTagListStruct:



Data Fields

- [gaiaExifTagPtr First](#)
pointer to first item into the linked list
- [gaiaExifTagPtr Last](#)
pointer to the last item into the linked list
- int [NumTags](#)
number of items
- [gaiaExifTagPtr * TagsArray](#)
an array of pointers to items

5.23.1 Detailed Description

Container for a list of EXIF tags.

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

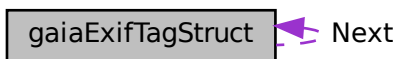
- `src/headers/spatialite/gaiaexif.h`

5.24 gaiaExifTagStruct Struct Reference

Container for an EXIF tag.

```
#include <gaiaexif.h>
```

Collaboration diagram for gaiaExifTagStruct:



Data Fields

- char [Gps](#)
GPS data included (0/1)
- unsigned short [TagId](#)
EXIF tag ID.
- unsigned short [Type](#)
EXIF value type.
- unsigned short [Count](#)
number of values
- unsigned char [TagOffset](#) [4]
tag offset [big- little-endian encoded]
- unsigned char * [ByteValue](#)
array of BYTE values
- char * [StringValue](#)
array of STRING values
- unsigned short * [ShortValues](#)
array of SHORT values
- unsigned int * [LongValues](#)
array of LONG values]
- unsigned int * [LongRationals1](#)
array of RATIONAL values [numerators]
- unsigned int * [LongRationals2](#)
array of RATIONAL values [denominators]
- short * [SignedShortValues](#)
array of Signed SHORT values
- int * [SignedLongValues](#)
array of Signed LONG values
- int * [SignedLongRationals1](#)
array of Signed RATIONAL values [numerators]
- int * [SignedLongRationals2](#)
array of Signed RATIONAL values [denominators]
- float * [FloatValues](#)
array of FLOAT values
- double * [DoubleValues](#)
array of DOUBLE values
- struct [gaiaExifTagStruct](#) * [Next](#)
pointer to next item into the linked list

5.24.1 Detailed Description

Container for an EXIF tag.

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

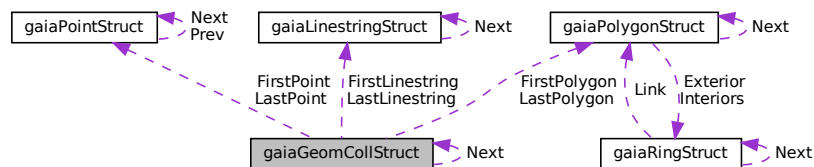
- `src/headers/spatialite/gaiaexif.h`

5.25 gaiaGeomCollStruct Struct Reference

Container for OGC GEOMETRYCOLLECTION Geometry.

```
#include <gg_structs.h>
```

Collaboration diagram for gaiaGeomCollStruct:



Data Fields

- `int Srid`
the SRID
- `char endian_arch`
CPU endian arch.
- `char endian`
BLOB Geometry endian arch.
- `const unsigned char * blob`
BLOB-Geometry buffer.
- `unsigned long size`
BLOB-Geometry buffer size (in bytes)
- `unsigned long offset`
current offset [BLOB parsing]
- `gaiaPointPtr FirstPoint`
pointer to first POINT [linked list]; may be NULL
- `gaiaPointPtr LastPoint`
pointer to last POINT [linked list]; may be NULL
- `gaiaLinestringPtr FirstLinestring`
pointer to first LINESTRING [linked list]; may be NULL
- `gaiaLinestringPtr LastLinestring`
pointer to last LINESTRING [linked list]; may be NULL
- `gaiaPolygonPtr FirstPolygon`

- pointer to first POLYGON [linked list]; may be NULL
 - [gaiaPolygonPtr LastPolygon](#)
- pointer to last POLYGON [linked list]; may be NULL
 - double [MinX](#)
 - MBR: min X.
 - double [MinY](#)
 - MBR: min Y.
 - double [MaxX](#)
 - MBR: max X.
 - double [MaxY](#)
 - MBR: max Y.
- int [DimensionModel](#)
- one of GAIA_XY, GAIA_XY_Z, GAIA_XY_M, GAIA_XY_ZM
- int [DeclaredType](#)
- any valid Geometry Class type
- struct [gaiaGeomCollStruct](#) * [Next](#)
- pointer to next item [linked list]

5.25.1 Detailed Description

Container for OGC GEOMETRYCOLLECTION Geometry.

Examples

[demo1.c](#), [demo2.c](#), [demo3.c](#), and [demo4.c](#).

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

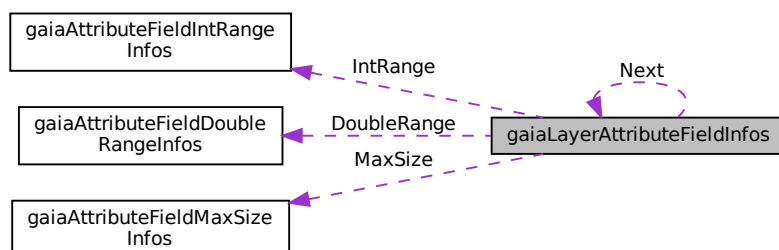
- [src/headers/spatialite/gg_structs.h](#)

5.26 gaiaLayerAttributeFieldInfos Struct Reference

LayerAttributeField infos.

```
#include <gg_structs.h>
```

Collaboration diagram for gaiaLayerAttributeFieldInfos:



Data Fields

- int [Ordinal](#)
ordinal position
- char * [AttributeFieldName](#)
SQL name of the corresponding column.
- int [NullValuesCount](#)
total count of NULL values
- int [IntegerValuesCount](#)
total count of INTEGER values
- int [DoubleValuesCount](#)
total count of DOUBLE values
- int [TextValuesCount](#)
total count of TEXT values
- int [BlobValuesCount](#)
total count of BLOB values
- [gaiaAttributeFieldMaxSizePtr](#) [MaxSize](#)
pointer to MaxSize/Length infos (may be NULL)
- [gaiaAttributeFieldIntRangePtr](#) [IntRange](#)
pointer to range of Integer values infos (may be NULL)
- [gaiaAttributeFieldDoubleRangePtr](#) [DoubleRange](#)
pointer to range of Double values infos (may be NULL)
- struct [gaiaLayerAttributeFieldInfos](#) * [Next](#)
pointer to next item (linked list)

5.26.1 Detailed Description

LayerAttributeField infos.

Examples

[demo5.c](#).

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

- [src/headers/spatialite/gg_structs.h](#)

5.27 gaiaLayerAuthInfos Struct Reference

Layer Auth infos.

```
#include <gg_structs.h>
```


Data Fields

- int [IsReadOnly](#)
Read-Only layer: TRUE or FALSE.
- int [IsHidden](#)
Hidden layer: TRUE or FALSE.
- int [HasTriggerInsert](#)
Flag indicating if the Capabilities of the SpatialView supports Inserting: TRUE or FALSE.
- int [HasTriggerUpdate](#)
Flag indicating if the Capabilities of the SpatialView supports Updating: TRUE or FALSE.
- int [HasTriggerDelete](#)
Flag indicating if the Capabilities of the SpatialView supports Deleting: TRUE or FALSE.

5.27.1 Detailed Description

Layer Auth infos.

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

- [src/headers/spatialite/gg_structs.h](#)

5.28 gaiaLayerExtentInfos Struct Reference

Layer Extent infos.

```
#include <gg_structs.h>
```

Data Fields

- int [Count](#)
row count (aka feature count)
- double [MinX](#)
Extent: min X.
- double [MinY](#)
Extent: min Y.
- double [MaxX](#)
Extent: max X.
- double [MaxY](#)
Extent: max Y.

5.28.1 Detailed Description

Layer Extent infos.

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

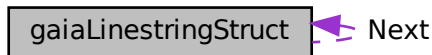
- [src/headers/spatialite/gg_structs.h](#)

5.29 gaiaLinestringStruct Struct Reference

Container for OGC LINESTRING Geometry.

```
#include <gg_structs.h>
```

Collaboration diagram for gaiaLinestringStruct:



Data Fields

- int [Points](#)
number of points [aka vertices]
- double * [Coords](#)
COORDs mem-array.
- double [MinX](#)
MBR: min X.
- double [MinY](#)
MBR: min Y.
- double [MaxX](#)
MBR: max X.
- double [MaxY](#)
MBR: max Y.
- int [DimensionModel](#)
one of GAIA_XY, GAIA_XY_Z, GAIA_XY_M, GAIA_XY_ZM
- struct [gaiaLinestringStruct](#) * [Next](#)
pointer to next item [linked list]

5.29.1 Detailed Description

Container for OGC LINESTRING Geometry.

Examples

[demo2.c](#).

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

- [src/headers/spatialite/gg_structs.h](#)

5.30 gaiaMemFileStruct Struct Reference

A Memory based File.

```
#include <gg_structs.h>
```

Data Fields

- char * **path**
- void * **buf**
- uint64_t **size**
- uint64_t **offset**

5.30.1 Detailed Description

A Memory based File.

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

- [src/headers/spatialite/gg_structs.h](#)

5.31 gaiaOutBufferStruct Struct Reference

Container for dynamically growing output buffer.

```
#include <gg_structs.h>
```

Data Fields

- char * [Buffer](#)
current buffer
- int [WriteOffset](#)
current write offset
- int [BufferSize](#)
current buffer size (in bytes)
- int [Error](#)
validity flag

5.31.1 Detailed Description

Container for dynamically growing output buffer.

Examples

[demo2.c](#).

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

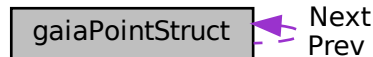
- [src/headers/spatialite/gg_structs.h](#)

5.32 gaiaPointStruct Struct Reference

Container for OGC POINT Geometry.

```
#include <gg_structs.h>
```

Collaboration diagram for gaiaPointStruct:



Data Fields

- double [X](#)
X coordinate.
- double [Y](#)
Y coordinate.
- double [Z](#)
Z coordinate: only for XYZ and XYZM dims.
- double [M](#)
M measure: only for XYM and XYZM dims.
- int [DimensionModel](#)
one of GAIA_XY, GAIA_XY_Z, GAIA_XY_M, GAIA_XY_ZM
- struct [gaiaPointStruct](#) * [Next](#)
pointer to next item [double linked list]
- struct [gaiaPointStruct](#) * [Prev](#)
pointer to previous item [double linked list]

5.32.1 Detailed Description

Container for OGC POINT Geometry.

Examples

[demo2.c](#).

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

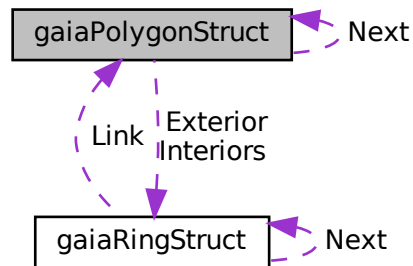
- [src/headers/spatialite/gg_structs.h](#)

5.33 gaiaPolygonStruct Struct Reference

Container for OGC POLYGON Geometry.

```
#include <gg_structs.h>
```

Collaboration diagram for gaiaPolygonStruct:



Data Fields

- [gaiaRingPtr Exterior](#)
the exterior ring (mandatory)
- int [NumInteriors](#)
number of interior rings (may be, none)
- [gaiaRingPtr Interiors](#)
array of interior rings
- int [NextInterior](#)
index of first unused interior ring
- double [MinX](#)
MBR: min X.
- double [MinY](#)
MBR: min Y.
- double [MaxX](#)
MBR: max X.
- double [MaxY](#)
MBR: max Y.
- int [DimensionModel](#)
one of GAIA_XY, GAIA_XY_Z, GAIA_XY_M, GAIA_XY_ZM
- struct [gaiaPolygonStruct](#) * [Next](#)
pointer to next item [linked list]

5.33.1 Detailed Description

Container for OGC POLYGON Geometry.

Examples

[demo2.c](#).

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

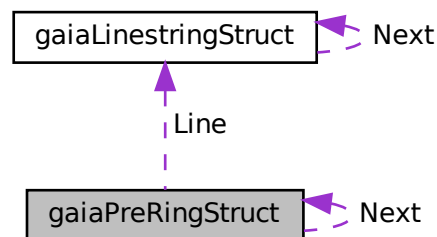
- [src/headers/spatialite/gg_structs.h](#)

5.34 gaiaPreRingStruct Struct Reference

Container similar to LINESTRING [internally used].

```
#include <gg_structs.h>
```

Collaboration diagram for gaiaPreRingStruct:



Data Fields

- [gaiaLinestringPtr](#) `Line`
pointer to LINESTRING
- `int` [AlreadyUsed](#)
already used/visited item
- `struct` [gaiaPreRingStruct](#) * `Next`
pointer to next item [linked list]

5.34.1 Detailed Description

Container similar to LINESTRING [internally used].

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

- [src/headers/spatialite/gg_structs.h](#)

5.35 gaiaProjAreaStr Struct Reference

BBOX corresponding to PROJ.6 AREA.

```
#include <gg_structs.h>
```

Data Fields

- double **WestLongitude**
- double **SouthLatitude**
- double **EastLongitude**
- double **NorthLatitude**

5.35.1 Detailed Description

BBOX corresponding to PROJ.6 AREA.

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

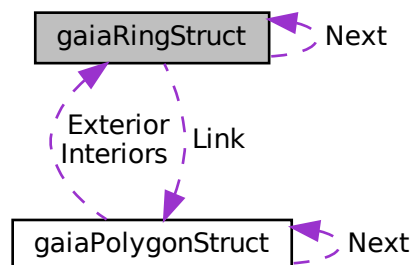
- [src/headers/spatialite/gg_structs.h](#)

5.36 gaiaRingStruct Struct Reference

Container for OGC RING Geometry.

```
#include <gg_structs.h>
```

Collaboration diagram for gaiaRingStruct:



Data Fields

- int [Points](#)
number of points [aka vertices]
- double * [Coords](#)
COORDs mem-array.
- int [Clockwise](#)
clockwise / counterclockwise
- double [MinX](#)
MBR: min X.
- double [MinY](#)
MBR: min Y.
- double [MaxX](#)
MBR: max X.
- double [MaxY](#)
MBR: max Y.
- int [DimensionModel](#)
one of GAIA_XY, GAIA_XY_Z, GAIA_XY_M, GAIA_XY_ZM
- struct [gaiaRingStruct](#) * [Next](#)
pointer to next item [linked list]
- struct [gaiaPolygonStruct](#) * [Link](#)
pointer to belonging Polygon

5.36.1 Detailed Description

Container for OGC RING Geometry.

Examples

[demo2.c](#).

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

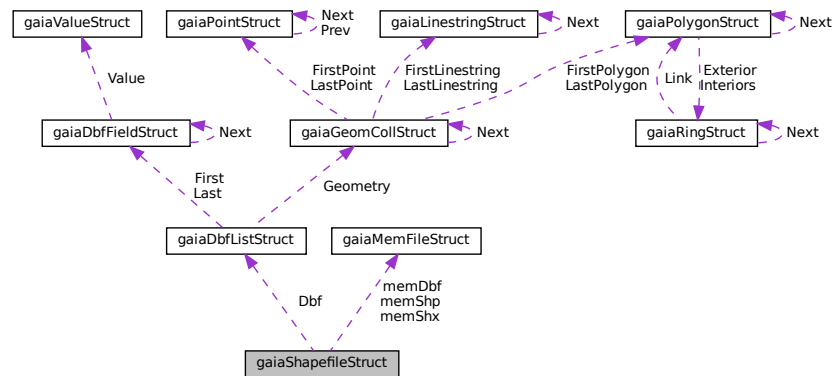
- [src/headers/spatialite/gg_structs.h](#)

5.37 gaiaShapefileStruct Struct Reference

Container for SHP file handling.

```
#include <gg_structs.h>
```


Collaboration diagram for gaiaShapefileStruct:



Data Fields

- int [endian_arch](#)
SHP endian arch.
- int [Valid](#)
validity flag: 1 = ready to be processed
- int [ReadOnly](#)
read or write mode
- char * [Path](#)
SHP 'abstract' path (no suffixes)
- FILE * [flShx](#)
FILE handle to SHX file.
- FILE * [flShp](#)
FILE handle to SHP file.
- FILE * [flDbf](#)
FILE handle to DBF file.
- [gaiaMemFilePtr](#) [memShx](#)
Memory based SHX file.
- [gaiaMemFilePtr](#) [memShp](#)
Memory based SHP file.
- [gaiaMemFilePtr](#) [memDbf](#)
Memory based DBF file.
- int [Shape](#)
the SHP shape code
- [gaiaDbfListPtr](#) [Dbf](#)
list of DBF fields
- unsigned char * [BufDbf](#)
DBF I/O buffer.
- int [DbfHdsz](#)
DBF header size (in bytes)
- int [DbfReclen](#)
DBF record length (in bytes)
- int [DbfSize](#)

- *DBF current file size (in bytes)*
- int [DbfRecno](#)
DBF current Record Number.
- unsigned char * [BufShp](#)
SHP I/O buffer.
- int [ShpBfsz](#)
SHP current buffer size (in bytes)
- int [ShpSize](#)
SHP current file size.
- int [ShxSize](#)
SHX current file size.
- double [MinX](#)
Total Extent: min X.
- double [MinY](#)
Total Extent: min Y.
- double [MaxX](#)
Total Extent: max X.
- double [MaxY](#)
Total Extent: max Y.
- void * [IconvObj](#)
handle to ICONV converter object
- char * [LastError](#)
last error message (may be NULL)
- int [EffectiveType](#)
SHP actual OGC Geometry type.
- int [EffectiveDims](#)
SHP actual dims: one of GAIA_XY, GAIA_XY_Z, GAIA_XY_M, GAIA_XY_ZM.

5.37.1 Detailed Description

Container for SHP file handling.

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

- [src/headers/spatialite/gg_structs.h](#)

5.38 gaiaValueStruct Struct Reference

Container for variant (multi-type) value.

```
#include <gg_structs.h>
```

Data Fields

- short [Type](#)
data type: one of GAIA_NULL_VALUE, GAIA_INT_VALUE, GAIA_DOUBLE_VALUE, GAIA_TEXT_VALUE
- char * [TxtValue](#)
TEXT type value.
- sqlite3_int64 [IntValue](#)
INT type value.
- double [DblValue](#)
DOUBLE type value.

5.38.1 Detailed Description

Container for variant (multi-type) value.

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

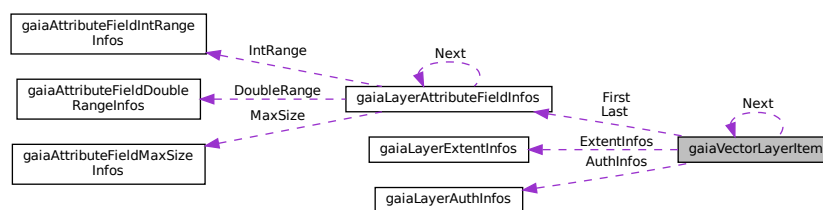
- `src/headers/spatialite/gg_structs.h`

5.39 gaiaVectorLayerItem Struct Reference

Vector Layer item.

```
#include <gg_structs.h>
```

Collaboration diagram for gaiaVectorLayerItem:



Data Fields

- `int LayerType`
one of `GAIA_VECTOR_UNKNOWN`, `GAIA_VECTOR_TABLE`, `GAIA_VECTOR_VIEW`, `GAIA_VECTOR_VIRTUAL`
- `char * TableName`
SQL name of the corresponding table.
- `char * GeometryName`
SQL name of the corresponding Geometry column.
- `int Srid`
SRID value.
- `int GeometryType`
one of `GAIA_VECTOR_UNKNOWN`, `GAIA_VECTOR_POINT`, `GAIA_VECTOR_LINestring`, `GAIA_VECTOR_POLYGON`, `GAIA_VECTOR_MULTIPoint`, `GAIA_VECTOR_MULTILINestring`, `GAIA_VECTOR_MULTIPOLYGON`, `GAIA_VECTOR_GEOMETRYCOLLECTION`, `GAIA_VECTOR_GEOMETRY`
- `int Dimensions`
one of `GAIA_VECTOR_UNKNOWN`, `GAIA_XY`, `GAIA_XY_Z`, `GAIA_XY_M`, `GAIA_XY_ZM`
- `int SpatialIndex`
one of `GAIA_VECTOR_UNKNOWN`, `GAIA_SPATIAL_INDEX_NONE`, `GAIA_SPATIAL_INDEX_RTREE`, `GAIA_SPATIAL_INDEX_MBRCACHE`
- `gaiaLayerExtentPtr ExtentInfos`
pointer to Extent infos (may be NULL)
- `gaiaLayerAuthPtr AuthInfos`
pointer to Auth infos (may be NULL)
- `gaiaLayerAttributeFieldPtr First`
pointer to first Field/Attribute (linked list)
- `gaiaLayerAttributeFieldPtr Last`
pointer to last Field/Attribute (linked list)
- `struct gaiaVectorLayerItem * Next`
pointer to next item (linked list)

5.39.1 Detailed Description

Vector Layer item.

Examples

[demo5.c](#).

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

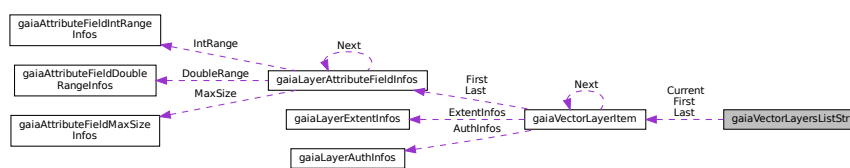
- [src/headers/spatialite/gg_structs.h](#)

5.40 gaiaVectorLayersListStr Struct Reference

Container for Vector Layers List.

```
#include <gg_structs.h>
```

Collaboration diagram for gaiaVectorLayersListStr:



Data Fields

- [gaiaVectorLayerPtr First](#)
pointer to first vector layer (linked list)
- [gaiaVectorLayerPtr Last](#)
pointer to last vector layer (linked list)
- [gaiaVectorLayerPtr Current](#)
pointer to currently set vector layer

5.40.1 Detailed Description

Container for Vector Layers List.

Examples

[demo5.c](#).

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

- [src/headers/spatialite/gg_structs.h](#)

5.41 vrttxt_column_header Struct Reference

Container for Virtual Text column (field) header.

```
#include <gg_structs.h>
```

Data Fields

- char * [name](#)
column name
- int [type](#)
data type: one of GAIA_NULL_VALUE, GAIA_INT_VALUE, GAIA_DOUBLE_VALUE, GAIA_TEXT_VALUE

5.41.1 Detailed Description

Container for Virtual Text column (field) header.

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

- [src/headers/spatialite/gg_structs.h](#)

5.42 vrttxt_line Struct Reference

Container for Virtual Text record (line)

```
#include <gg_structs.h>
```

Data Fields

- [gaia_off_t offset](#)
current offset (parsing)
- int [len](#)
line length (in bytes)
- int [field_offsets](#) [[VRTTXT_FIELDS_MAX](#)]
array of field offsets (where each field starts)
- int [num_fields](#)
number of field into the record
- int [error](#)
validity flag

5.42.1 Detailed Description

Container for Virtual Text record (line)

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

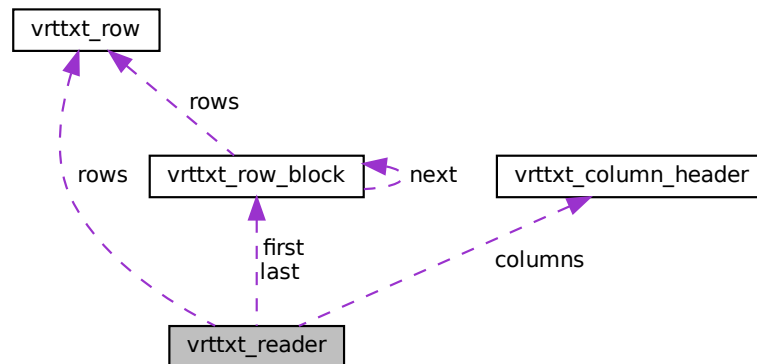
- [src/headers/spatialite/gg_structs.h](#)

5.43 vrttxt_reader Struct Reference

Container for Virtual Text file handling.

```
#include <gg_structs.h>
```

Collaboration diagram for vrttxt_reader:



Data Fields

- struct `vrttxt_column_header` `columns` [`VRTTXT_FIELDS_MAX`]
array of columns (fields)
- FILE * `text_file`
FILE handle.
- void * `toUtf8`
handle to ICONV converter object
- char `field_separator`
field separator character
- char `text_separator`
text separator character (quote)
- char `decimal_separator`
decimal separator
- int `first_line_titles`
TRUE if the first line contains column names.
- int `error`
validity flag
- struct `vrttxt_row_block` * `first`
pointer to first block of records [linked list]
- struct `vrttxt_row_block` * `last`
pointer to last block of records [linked list]
- struct `vrttxt_row` ** `rows`
array of pointers to individual records [lines]
- int `num_rows`
number of records

- int [line_no](#)
current Line Number
- int [max_fields](#)
max number of columns (fields)
- int [current_buf_sz](#)
current buffer size
- int [current_buf_off](#)
current buffer offset [parsing]
- char * [line_buffer](#)
I/O buffer.
- char * [field_buffer](#)
current field buffer
- int [field_offsets](#) [VRTTXT_FIELDS_MAX]
array of field offsets [current record]
- int [field_lens](#) [VRTTXT_FIELDS_MAX]
array of field lengths [current record]
- int [max_current_field](#)
max field [current record]
- int [current_line_ready](#)
current record [line] ready for parsing

5.43.1 Detailed Description

Container for Virtual Text file handling.

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

- [src/headers/spatialite/gg_structs.h](#)

5.44 vrttxt_row Struct Reference

Container for Virtual Text record (line) offsets.

```
#include <gg_structs.h>
```

Data Fields

- int [line_no](#)
Line Number.
- [gaia_off_t](#) [offset](#)
start offset
- int [len](#)
record (line) length (in bytes)
- int [num_fields](#)
number of fields into this record

5.44.1 Detailed Description

Container for Virtual Text record (line) offsets.

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

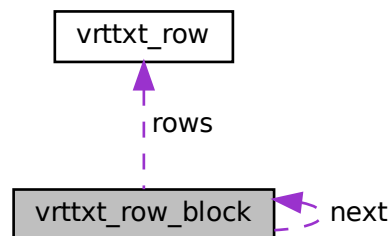
- [src/headers/spatialite/gg_structs.h](#)

5.45 vrttxt_row_block Struct Reference

Container for Virtual Text block of records.

```
#include <gg_structs.h>
```

Collaboration diagram for vrttxt_row_block:



Data Fields

- struct [vrttxt_row](#) [rows](#) [[VRTTXT_BLOCK_MAX](#)]
array of records [lines]
- int [num_rows](#)
number of records into the array
- int [min_line_no](#)
min Line Number
- int [max_line_no](#)
max Line Number
- struct [vrttxt_row_block](#) * [next](#)
pointer to next item [linked list]

5.45.1 Detailed Description

Container for Virtual Text block of records.

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

- [src/headers/spatialite/gg_structs.h](#)

Chapter 6

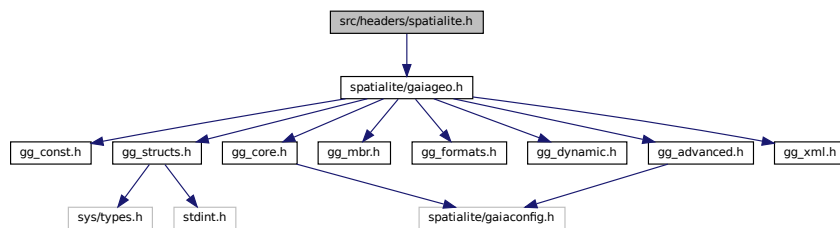
File Documentation

6.1 src/headers/spatialite.h File Reference

Main Spatialite header file.

```
#include <spatialite/gaiageo.h>
```

Include dependency graph for spatialite.h:



Macros

- `#define SPLITE_AXIS_1 0x51`
- `#define SPLITE_AXIS_2 0x52`
- `#define SPLITE_AXIS_NAME 0x3e`
- `#define SPLITE_AXIS_ORIENTATION 0x3f`

Functions

- SPATIALITE_DECLARE void [spatialite_initialize](#) (void)
Initializes the library.
- SPATIALITE_DECLARE void [spatialite_shutdown](#) (void)
Finalizes the library.
- SPATIALITE_DECLARE const char * [spatialite_version](#) (void)
Return the current library version.
- SPATIALITE_DECLARE const char * [spatialite_target_cpu](#) (void)

- Return the target CPU name.*

 - SPATIALITE_DECLARE void * [spatialite_alloc_connection](#) (void)

Initializes the internal memory block supporting each connection.
- SPATIALITE_DECLARE void [spatialite_set_silent_mode](#) (const void *ptr)

Disables reporting GEOS/RTTOPO error and warnings on stderr.
- SPATIALITE_DECLARE void [spatialite_set_verbode_mode](#) (const void *ptr)

Enables reporting GEOS/RTTOPO error and warnings on stderr.
- SPATIALITE_DECLARE void [spatialite_init](#) (int verbose)

Initializes a Spatialite connection.
- SPATIALITE_DECLARE void [spatialite_init_ex](#) (sqlite3 *db_handle, const void *ptr, int verbose)

Initializes a Spatialite connection.
- SPATIALITE_DECLARE void [spatialite_init_geos](#) (void)

Initializes the GEOS library.
- SPATIALITE_DECLARE void [spatialite_cleanup](#) (void)

Cleanup a Spatialite connection.
- SPATIALITE_DECLARE void [spatialite_cleanup_ex](#) (const void *ptr)

Cleanup a Spatialite connection.
- SPATIALITE_DECLARE void [spatialite_finalize_topologies](#) (const void *ptr)

Partially Cleaning-up a Spatialite connection.
- SPATIALITE_DECLARE void [enable_tiny_point](#) (const void *ptr)

Enables the BLOB-TinyPoint encoding.
- SPATIALITE_DECLARE void [disable_tiny_point](#) (const void *ptr)

Disables the BLOB-TinyPoint encoding.
- SPATIALITE_DECLARE int [is_tiny_point_enabled](#) (const void *ptr)

Checks if the BLOB-TinyPoint encoding is enabled or not.
- SPATIALITE_DECLARE int [dump_shapefile](#) (sqlite3 *sqlite, char *table, char *column, char *shp_path, char *charset, char *geom_type, int verbose, int *rows, char *err_msg)

Dumps a full geometry-table into an external Shapefile.
- SPATIALITE_DECLARE int [dump_shapefile_ex](#) (sqlite3 *sqlite, char *table, char *column, char *shp_path, char *charset, char *geom_type, int verbose, int *rows, int colcase_name, char *err_msg)

Dumps a full geometry-table into an external Shapefile.
- SPATIALITE_DECLARE int [dump_shapefile_ex2](#) (sqlite3 *sqlite, void *proj_ctx, char *table, char *column, char *shp_path, char *charset, char *geom_type, int verbose, int *rows, int colcase_name, char *err_msg)

Dumps a full geometry-table into an external Shapefile.
- SPATIALITE_DECLARE int [load_shapefile](#) (sqlite3 *sqlite, char *shp_path, char *table, char *charset, int srid, char *column, int coerce2d, int compressed, int verbose, int spatial_index, int *rows, char *err_msg)

Loads an external Shapefile into a newly created table.
- SPATIALITE_DECLARE int [load_shapefile_ex](#) (sqlite3 *sqlite, char *shp_path, char *table, char *charset, int srid, char *geo_column, char *gtype, char *pk_column, int coerce2d, int compressed, int verbose, int spatial_index, int *rows, char *err_msg)

Loads an external Shapefile into a newly created table.
- SPATIALITE_DECLARE int [load_shapefile_ex2](#) (sqlite3 *sqlite, char *shp_path, char *table, char *charset, int srid, char *geo_column, char *gtype, char *pk_column, int coerce2d, int compressed, int verbose, int spatial_index, int text_date, int *rows, char *err_msg)

Loads an external Shapefile into a newly created table.
- SPATIALITE_DECLARE int [load_shapefile_ex3](#) (sqlite3 *sqlite, const char *shp_path, const char *table, const char *charset, int srid, const char *geo_column, const char *gtype, const char *pk_column, int coerce2d, int compressed, int verbose, int spatial_index, int text_date, int *rows, int colname_case, char *err←_msg)

Loads an external Shapefile into a newly created table.

- SPATIALITE_DECLARE int [load_zip_shapefile](#) (sqlite3 *sqlite, const char *zip_path, const char *shp_path, const char *table, const char *charset, int srid, const char *geo_column, const char *gtype, const char *pk_column, int coerce2d, int compressed, int verbose, int spatial_index, int text_date, int *rows, int colname_case, char *err_msg)
Loads an external Shapefile (from Zipfile) into a newly created table.
- SPATIALITE_DECLARE int [load_dbf](#) (sqlite3 *sqlite, char *dbf_path, char *table, char *charset, int verbose, int *rows, char *err_msg)
Loads an external DBF file into a newly created table.
- SPATIALITE_DECLARE int [load_dbf_ex](#) (sqlite3 *sqlite, char *dbf_path, char *table, char *pk_column, char *charset, int verbose, int *rows, char *err_msg)
Loads an external DBF file into a newly created table.
- SPATIALITE_DECLARE int [load_dbf_ex2](#) (sqlite3 *sqlite, char *dbf_path, char *table, char *pk_column, char *charset, int verbose, int text_date, int *rows, char *err_msg)
Loads an external DBF file into a newly created table.
- SPATIALITE_DECLARE int [load_dbf_ex3](#) (sqlite3 *sqlite, const char *dbf_path, const char *table, const char *pk_column, const char *charset, int verbose, int text_date, int *rows, int colname_case, char *err_msg)
Loads an external DBF file into a newly created table.
- SPATIALITE_DECLARE int [load_zip_dbf](#) (sqlite3 *sqlite, const char *zip_file, const char *dbf_path, const char *table, const char *pk_column, const char *charset, int verbose, int text_date, int *rows, int colname_case, char *err_msg)
Loads an external DBF file (from Zipfile) into a newly created table.
- SPATIALITE_DECLARE int [dump_dbf](#) (sqlite3 *sqlite, char *table, char *dbf_path, char *charset, char *err_msg)
Dumps a full table into an external DBF file.
- SPATIALITE_DECLARE int [dump_dbf_ex](#) (sqlite3 *sqlite, char *table, char *dbf_path, char *charset, int *rows, char *err_msg)
Dumps a full table into an external DBF file.
- SPATIALITE_DECLARE int [dump_dbf_ex2](#) (sqlite3 *sqlite, char *table, char *dbf_path, char *charset, int *rows, int colname_case, char *err_msg)
Dumps a full table into an external DBF file.
- SPATIALITE_DECLARE int [load_XL](#) (sqlite3 *sqlite, const char *path, const char *table, unsigned int worksheetIndex, int first_titles, unsigned int *rows, char *err_msg)
Loads an external spreadsheet (.xls) file into a newly created table.
- SPATIALITE_DECLARE double [math_round](#) (double value)
A portable replacement for C99 round()
- SPATIALITE_DECLARE sqlite3_int64 [math_llabs](#) (sqlite3_int64 value)
A portable replacement for C99 llabs()
- SPATIALITE_DECLARE int [spatial_ref_sys_init](#) (sqlite3 *sqlite, int verbose)
Inserts the inlined EPSG dataset into the "spatial_ref_sys" table.
- SPATIALITE_DECLARE int [spatial_ref_sys_init2](#) (sqlite3 *sqlite, int mode, int verbose)
Inserts the inlined EPSG dataset into the "spatial_ref_sys" table.
- SPATIALITE_DECLARE int [insert_epsg_srid](#) (sqlite3 *sqlite, int srid)
Inserts some inlined EPSG definition into the "spatial_ref_sys" table.
- SPATIALITE_DECLARE int [srid_is_geographic](#) (sqlite3 *sqlite, int srid, int *geographic)
checks a SRID definition from the "spatial_ref_sys" table determining if it is of the geographic type
- SPATIALITE_DECLARE int [srid_is_projected](#) (sqlite3 *sqlite, int srid, int *projected)
checks a SRID definition from the "spatial_ref_sys" table determining if it is of the projected type
- SPATIALITE_DECLARE int [srid_has_flipped_axes](#) (sqlite3 *sqlite, int srid, int *flipped)
checks a SRID definition from the "spatial_ref_sys" table determining if the axes order is X-Y or Y-X
- SPATIALITE_DECLARE char * [srid_get_spheroid](#) (sqlite3 *sqlite, int srid)
checks a SRID definition from the "spatial_ref_sys" table then returning the corresponding Spheroid name
- SPATIALITE_DECLARE char * [srid_get_prime_meridian](#) (sqlite3 *sqlite, int srid)

- checks a SRID definition from the "spatial_ref_sys" table then returning the corresponding Prime Meridian name*

 - SPATIALITE_DECLARE char * [srid_get_projection](#) (sqlite3 *sqlite, int srid)
- checks a SRID definition from the "spatial_ref_sys" table then returning the corresponding Projection name*

 - SPATIALITE_DECLARE char * [srid_get_datum](#) (sqlite3 *sqlite, int srid)
- checks a SRID definition from the "spatial_ref_sys" table then returning the corresponding Datum name*

 - SPATIALITE_DECLARE char * [srid_get_unit](#) (sqlite3 *sqlite, int srid)
- checks a SRID definition from the "spatial_ref_sys" table then returning the corresponding Unit name*

 - SPATIALITE_DECLARE char * [srid_get_axis](#) (sqlite3 *sqlite, int srid, char axis, char mode)
- checks a SRID definition from the "spatial_ref_sys" table then returning an Axis definition*

 - SPATIALITE_DECLARE int [is_kml_constant](#) (sqlite3 *sqlite, char *table, char *column)
- Checks if a column is actually defined into the given table.*

 - SPATIALITE_DECLARE int [dump_kml](#) (sqlite3 *sqlite, char *table, char *geom_col, char *kml_path, char *name_col, char *desc_col, int precision)
- Dumps a full geometry-table into an external KML file.*

 - SPATIALITE_DECLARE int [dump_kml_ex](#) (sqlite3 *sqlite, char *table, char *geom_col, char *kml_path, char *name_col, char *desc_col, int precision, int *rows)
- Dumps a full geometry-table into an external KML file.*

 - SPATIALITE_DECLARE void [check_duplicated_rows](#) (sqlite3 *sqlite, char *table, int *dupl_count)
- Checks for duplicated rows into the same table.*

 - SPATIALITE_DECLARE void [remove_duplicated_rows](#) (sqlite3 *sqlite, char *table)
- Remove duplicated rows from a table.*

 - SPATIALITE_DECLARE void [remove_duplicated_rows_ex](#) (sqlite3 *sqlite, char *table, int *removed)
- Remove duplicated rows from a table.*

 - SPATIALITE_DECLARE void [remove_duplicated_rows_ex2](#) (sqlite3 *sqlite, char *table, int *removed, int transaction)
- Remove duplicated rows from a table.*

 - SPATIALITE_DECLARE void [elementary_geometries](#) (sqlite3 *sqlite, char *inTable, char *geometry, char *outTable, char *pKey, char *multild)
- Creates a derived table surely containing elementary Geometries.*

 - SPATIALITE_DECLARE void [elementary_geometries_ex](#) (sqlite3 *sqlite, char *inTable, char *geometry, char *outTable, char *pKey, char *multild, int *rows)
- Creates a derived table surely containing elementary Geometries.*

 - SPATIALITE_DECLARE void [elementary_geometries_ex2](#) (sqlite3 *sqlite, char *inTable, char *geometry, char *outTable, char *pKey, char *multild, int *rows, int transaction)
- Creates a derived table surely containing elementary Geometries.*

 - SPATIALITE_DECLARE void [elementary_geometries_ex3](#) (sqlite3 *sqlite, char *inTable, char *geometry, char *outTable, char *pKey, char *multild, const void *options, int *rows, int transaction)
- Creates a derived table surely containing elementary Geometries.*

 - SPATIALITE_DECLARE int [dump_geojson](#) (sqlite3 *sqlite, char *table, char *geom_col, char *outfile_path, int precision, int option)
- Dumps a full geometry-table into an external GeoJSON file (old specification)*

 - SPATIALITE_DECLARE int [dump_geojson_ex](#) (sqlite3 *sqlite, char *table, char *geom_col, char *outfile_path, int precision, int option, int *rows)
- Dumps a full geometry-table into an external GeoJSON file (old specification)*

 - SPATIALITE_DECLARE int [dump_geojson2](#) (sqlite3 *sqlite, char *table, char *geom_col, char *outfile_path, int precision, int lon_lat, int m_coords, int indented, int colname_case, int *rows, char **error_message)
- Dumps a full geometry-table into an external GeoJSON file (RFC 7946)*

 - SPATIALITE_DECLARE int [load_geojson](#) (sqlite3 *sqlite, char *path, char *table, char *column, int spatial_index, int srid, int colname_case, int *rows, char **error_message)
- Loads an external GeoJSON file into a newly created table.*

 - SPATIALITE_DECLARE int [update_layer_statistics](#) (sqlite3 *sqlite, const char *table, const char *column)
- Updates the LAYER_STATICS metadata table.*

- SPATIALITE_DECLARE int [gaiaStatisticsInvalidate](#) (sqlite3 *handle, const char *table, const char *geometry)
Immediately and unconditionally invalidates the already existing Statistics.
- SPATIALITE_DECLARE [gaiaGeomCollPtr](#) [gaiaGetLayerExtent](#) (sqlite3 *handle, const char *table, const char *geometry, int mode)
Queries the Metadata tables returning the Layer Full Extent.
- SPATIALITE_DECLARE [gaiaVectorLayersListPtr](#) [gaiaGetVectorLayersList](#) (sqlite3 *handle, const char *table, const char *geometry, int mode)
Queries the Metadata tables supporting Vector Layers.
- SPATIALITE_DECLARE int [gaiaCreateMetaCatalogTables](#) (sqlite3 *handle)
Creates (or re-creates) the "splite_metacatalog" and "splite_metacatalog_statistics" tables.
- SPATIALITE_DECLARE int [gaiaUpdateMetaCatalogStatistics](#) (sqlite3 *handle, const char *table, const char *column)
Updates the "splite_metacatalog_statistics" table.
- SPATIALITE_DECLARE int [gaiaUpdateMetaCatalogStatisticsFromMaster](#) (sqlite3 *handle, const char *master_table, const char *table_name, const char *column_name)
Updates the "splite_metacatalog_statistics" table (using a Master Table).
- SPATIALITE_DECLARE void [gaiaFreeVectorLayersList](#) ([gaiaVectorLayersListPtr](#) ptr)
Destroys a VectorLayersList object.
- SPATIALITE_DECLARE int [gaiaDropTable](#) (sqlite3 *sqlite, const char *table)
Drops a layer-table, removing any related dependency.
- SPATIALITE_DECLARE int [gaiaDropTableEx](#) (sqlite3 *sqlite, const char *prefix, const char *table)
Drops a layer-table, removing any related dependency.
- SPATIALITE_DECLARE int [gaiaDropTableEx2](#) (sqlite3 *sqlite, const char *prefix, const char *table, int transaction)
Drops a layer-table, removing any related dependency.
- SPATIALITE_DECLARE int [gaiaDropTableEx3](#) (sqlite3 *sqlite, const char *prefix, const char *table, int transaction, char **error_message)
Drops a layer-table, removing any related dependency.
- SPATIALITE_DECLARE int [gaiaDropTable5](#) (sqlite3 *sqlite, const char *prefix, const char *table, char **error_message)
Drops a layer-table, removing any related dependency.
- SPATIALITE_DECLARE int [gaiaRenameTable](#) (sqlite3 *sqlite, const char *prefix, const char *old_name, const char *new_name, char **error_message)
Renames a Table.
- SPATIALITE_DECLARE int [gaiaRenameColumn](#) (sqlite3 *sqlite, const char *prefix, const char *table, const char *old_name, const char *new_name, char **error_message)
Renames a Table's Column.
- SPATIALITE_DECLARE int [check_geometry_column](#) (sqlite3 *sqlite, const char *table, const char *geom, const char *report_path, int *n_rows, int *n_invalids, char **err_msg)
Checks a Geometry Column for validity.
- SPATIALITE_DECLARE int [check_geometry_column_r](#) (const void *p_cache, sqlite3 *sqlite, const char *table, const char *geom, const char *report_path, int *n_rows, int *n_invalids, char **err_msg)
Checks a Geometry Column for validity.
- SPATIALITE_DECLARE int [check_all_geometry_columns](#) (sqlite3 *sqlite, const char *output_dir, int *n_invalids, char **err_msg)
Checks all Geometry Columns for validity.
- SPATIALITE_DECLARE int [check_all_geometry_columns_r](#) (const void *p_cache, sqlite3 *sqlite, const char *output_dir, int *n_invalids, char **err_msg)
Checks all Geometry Columns for validity.
- SPATIALITE_DECLARE int [sanitize_geometry_column](#) (sqlite3 *sqlite, const char *table, const char *geom, const char *tmp_table, const char *report_path, int *n_invalids, int *n_repaired, int *n_discarded, int *n_failures, char **err_msg)
Sanitizes a Geometry Column making all invalid geometries to be valid.

- SPATIALITE_DECLARE int [sanitize_geometry_column_r](#) (const void *p_cache, sqlite3 *sqlite, const char *table, const char *geom, const char *tmp_table, const char *report_path, int *n_invalids, int *n_repaired, int *n_discarded, int *n_failures, char **err_msg)
Sanitizes a Geometry Column making all invalid geometries to be valid.
- SPATIALITE_DECLARE int [sanitize_all_geometry_columns](#) (sqlite3 *sqlite, const char *tmp_prefix, const char *output_dir, int *not_repaired, char **err_msg)
Sanitizes all Geometry Columns making all invalid geometries to be valid.
- SPATIALITE_DECLARE int [sanitize_all_geometry_columns_r](#) (const void *p_cache, sqlite3 *sqlite, const char *tmp_prefix, const char *output_dir, int *not_repaired, char **err_msg)
Sanitizes all Geometry Columns making all invalid geometries to be valid.
- SPATIALITE_DECLARE int [gaiaCutter](#) (sqlite3 *db_handle, const void *cache, const char *in_db_prefix, const char *input_table, const char *input_geom, const char *blade_db_prefix, const char *blade_table, const char *blade_geom, const char *output_table, int transaction, int ram_tmp_store, char **message)
Will precisely cut the input dataset against polygonal blade(s) and will consequently create and populate an output dataset.
- SPATIALITE_DECLARE int [gaia_create_routing_nodes](#) (sqlite3 *db_handle, const void *cache, const char *prefix, const char *table, const char *geom_column, const char *from_column, const char *to_column)
Will attempt to create the Routing Nodes columns for a spatial table.
- SPATIALITE_DECLARE int [gaia_create_routing](#) (sqlite3 *db_handle, const void *cache, const char *routing_data_table, const char *virtual_routing_table, const char *input_table, const char *from_column, const char *to_column, const char *geom_column, const char *cost_column, const char *name_column, int a_star_enabled, int bidirectional, const char *oneway_from, const char *oneway_to, int overwrite)
Will attempt to create a VirtualRouting from an input table.
- SPATIALITE_DECLARE const char * [gaia_create_routing_get_last_error](#) (const void *cache)
- SPATIALITE_DECLARE int [gaiaGPKG2Spatialite](#) (sqlite3 *handle_in, const char *gpkg_in_path, sqlite3 *handle_out, const char *splite_out_path)
- SPATIALITE_DECLARE int [gaiaSpatialite2GPKG](#) (sqlite3 *handle_in, const char *splite_in_path, sqlite3 *handle_out, const char *gpkg_out_path)
- SPATIALITE_DECLARE const void * [gaiaGetCurrentProjContext](#) (const void *cache)
- SPATIALITE_DECLARE int [gaiaSetCurrentCachedProj](#) (const void *cache, void *pj, const char *proj_string_1, const char *proj_string_2, void *area)
- SPATIALITE_DECLARE void * [gaiaGetCurrentCachedProj](#) (const void *cache)
- SPATIALITE_DECLARE int [gaiaCurrentCachedProjMatches](#) (const void *cache, const char *proj_string_1, const char *proj_string_2, void *area)
- SPATIALITE_DECLARE char * [gaiaGetDbObjectScope](#) (sqlite3 *handle, const char *db_prefix, const char *obj_name)

6.1.1 Detailed Description

Main Spatialite header file.

6.1.2 Function Documentation

6.1.2.1 check_all_geometry_columns()

```
SPATIALITE_DECLARE int check_all_geometry_columns (
    sqlite3 * sqlite,
    const char * output_dir,
    int * n_invalids,
    char ** err_msg )
```

Checks all Geometry Columns for validity.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>output_dir</i>	pathname of the directory to be created for report-files
<i>n_invalids</i>	if this variable is not NULL on successful completion will contain the total number of invalid Geometries found
<i>err_msg</i>	if this variable is not NULL and the return status is ZERO (failure), an appropriate error message will be returned

See also

[check_all_geometry_columns_r](#), [check_geometry_column](#), [sanitize_geometry_column](#), [sanitize_all_geometry_columns](#)

Note

this function will check all Geometry Columns (vector layers) for validity; a HTML report will be produced.
 an eventual error message returned via *err_msg* requires to be deallocated by invoking `free()`
 not reentrant and thread unsafe.

Returns

0 on failure, any other value on success

6.1.2.2 `check_all_geometry_columns_r()`

```
SPATIALITE_DECLARE int check_all_geometry_columns_r (
    const void * p_cache,
    sqlite3 * sqlite,
    const char * output_dir,
    int * n_invalids,
    char ** err_msg )
```

Checks all Geometry Columns for validity.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>sqlite</i>	handle to current DB connection
<i>output_dir</i>	pathname of the directory to be created for report-files
<i>n_invalids</i>	if this variable is not NULL on successful completion will contain the total number of invalid Geometries found
<i>err_msg</i>	if this variable is not NULL and the return status is ZERO (failure), an appropriate error message will be returned

See also

[check_all_geometry_columns](#), [check_geometry_column](#), [sanitize_geometry_column](#), [sanitize_all_geometry_columns](#)

Note

this function will check all Geometry Columns (vector layers) for validity; a HTML report will be produced. an eventual error message returned via `err_msg` requires to be deallocated by invoking `free()` reentrant and thread-safe.

Returns

0 on failure, any other value on success

6.1.2.3 check_duplicated_rows()

```
SPATIALITE_DECLARE void check_duplicated_rows (
    sqlite3 * sqlite,
    char * table,
    int * dupl_count )
```

Checks for duplicated rows into the same table.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>table</i>	name of the table to be checked
<i>dupl_count</i>	on completion will contain the number of duplicated rows found

See also

[remove_duplicated_rows](#)

Note

two (or more) rows are assumed to be duplicated if any column

value (excluding any Primary Key column) is exactly the same

6.1.2.4 check_geometry_column()

```
SPATIALITE_DECLARE int check_geometry_column (
    sqlite3 * sqlite,
    const char * table,
    const char * geom,
    const char * report_path,
    int * n_rows,
    int * n_invalids,
    char ** err_msg )
```

Checks a Geometry Column for validity.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>table</i>	name of the table
<i>geometry</i>	name of the column to be checked
<i>report_path</i>	pathname of the report-file
<i>n_rows</i>	if this variable is not NULL on successful completion will contain the total number of rows found into the checkeck table
<i>n_invalids</i>	if this variable is not NULL on successful completion will contain the total number of invalid Geometries found into the checkeck table
<i>err_msg</i>	if this variable is not NULL and the return status is ZERO (failure), an appropriate error message will be returned

See also

[check_geometry_column_r](#), [check_all_geometry_columns](#), [sanitize_geometry_column](#), [sanitize_all_geometry_columns](#)

Note

this function will check a Geometry Column (layer) for validity; a HTML report will be produced.
an eventual error message returned via *err_msg* requires to be deallocated by invoking `free()`
not reentrant and thread unsafe.

Returns

0 on failure, any other value on success

6.1.2.5 check_geometry_column_r()

```
SPATIALITE_DECLARE int check_geometry_column_r (
    const void * p_cache,
    sqlite3 * sqlite,
    const char * table,
    const char * geom,
    const char * report_path,
    int * n_rows,
    int * n_invalids,
    char ** err_msg )
```

Checks a Geometry Column for validity.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>sqlite</i>	handle to current DB connection
<i>table</i>	name of the table
<i>geometry</i>	name of the column to be checked
<i>report_path</i>	pathname of the report-file
<i>n_rows</i>	if this variable is not NULL on successful completion will contain the total number of rows found into the checkeck table
<i>n_invalids</i>	if this variable is not NULL on successful completion will contain the total number of invalid Geometries found into the checkeck table
<i>err_msg</i>	if this variable is not NULL and the return status is ZERO (failure), an appropriate error message will be returned

See also

[check_geometry_column](#), [check_all_geometry_columns](#), [sanitize_geometry_column](#), [sanitize_all_geometry_columns](#)

Note

this function will check a Geometry Column (layer) for validity; a HTML report will be produced. an eventual error message returned via `err_msg` requires to be deallocated by invoking `free()` reentrant and thread-safe.

Returns

0 on failure, any other value on success

6.1.2.6 disable_tiny_point()

```
SPATIALITE_DECLARE void disable_tiny_point (
    const void * ptr )
```

Disables the BLOB-TinyPoint encoding.

Parameters

<i>ptr</i>	the same memory pointer passed to the corresponding call to spatialite_init_ex() and returned by spatialite_alloc_connection()
------------	--

See also

[disable_tiny_point](#), [is_tiny_point_enabled](#)

6.1.2.7 dump_dbf()

```
SPATIALITE_DECLARE int dump_dbf (
    sqlite3 * sqlite,
    char * table,
    char * dbf_path,
    char * charset,
    char * err_msg )
```

Dumps a full table into an external DBF file.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>table</i>	the name of the table to be exported
<i>dbf_path</i>	pathname of the DBF to be exported
<i>charset</i>	a valid GNU ICONV charset to be used for DBF text strings
<i>err_msg</i>	on completion will contain an error message (if any)

See also

[dump_dbf_ex](#), [dump_dbf_ex2](#)

Returns

0 on failure, any other value on success

6.1.2.8 dump_dbf_ex()

```
SPATIALITE_DECLARE int dump_dbf_ex (
    sqlite3 * sqlite,
    char * table,
    char * dbf_path,
    char * charset,
    int * rows,
    char * err_msg )
```

Dumps a full table into an external DBF file.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>table</i>	the name of the table to be exported
<i>dbf_path</i>	pathname of the DBF to be exported
<i>charset</i>	a valid GNU ICONV charset to be used for DBF text strings
<i>rows</i>	on completion will contain the total number of exported rows
<i>err_msg</i>	on completion will contain an error message (if any)

See also

[dump_dbf](#), [dump_dbf_ex2](#)

Returns

0 on failure, any other value on success

6.1.2.9 dump_dbf_ex2()

```
SPATIALITE_DECLARE int dump_dbf_ex2 (
    sqlite3 * sqlite,
    char * table,
    char * dbf_path,
    char * charset,
    int * rows,
    int colname_case,
    char * err_msg )
```

Dumps a full table into an external DBF file.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>table</i>	the name of the table to be exported
<i>dbf_path</i>	pathname of the DBF to be exported
<i>charset</i>	a valid GNU ICONV charset to be used for DBF text strings
<i>rows</i>	on completion will contain the total number of exported rows
<i>colname_case</i>	one between GAIA_DBF_COLNAME_LOWERCASE, GAIA_DBF_COLNAME_UPPERCASE or GAIA_DBF_COLNAME_CASE_IGNORE.
<i>err_msg</i>	on completion will contain an error message (if any)

See also

[dump_dbf](#), [dump_dbf_ex](#)

Returns

0 on failure, any other value on success

6.1.2.10 dump_geojson()

```
SPATIALITE_DECLARE int dump_geojson (
    sqlite3 * sqlite,
    char * table,
    char * geom_col,
    char * outfile_path,
    int precision,
    int option )
```

Dumps a full geometry-table into an external GeoJSON file (old specification)

Parameters

<i>sqlite</i>	handle to current DB connection
<i>table</i>	the name of the table to be exported
<i>geom_col</i>	the name of the geometry column
<i>outfile_path</i>	pathname for the GeoJSON file to be written to
<i>precision</i>	number of decimal digits for coordinates
<i>option</i>	the format to use for output

See also

[dump_geojson_ex](#), [dump_geojson2](#)

Note

valid values for option are:

- 0 no option
- 1 GeoJSON MBR
- 2 GeoJSON Short CRS (e.g EPSG:4326)
- 3 MBR + Short CRS
- 4 GeoJSON Long CRS (e.g urn:ogc:def:crs:EPSG::4326)
- 5 MBR + Long CRS

Returns

0 on failure, any other value on success

6.1.2.11 dump_geojson2()

```
SPATIALITE_DECLARE int dump_geojson2 (
    sqlite3 * sqlite,
    char * table,
    char * geom_col,
    char * outfile_path,
    int precision,
    int lon_lat,
    int m_coords,
    int indented,
    int colname_case,
    int * rows,
    char ** error_message )
```

Dumps a full geometry-table into an external GeoJSON file (RFC 7946)

Parameters

<i>sqlite</i>	handle to current DB connection
<i>table</i>	the name of the table to be exported
<i>geom_col</i>	the name of the geometry column
<i>outfile_path</i>	pathname for the GeoJSON file to be written to
<i>precision</i>	number of decimal digits for coordinates
<i>lon_lat</i>	TRUE if all coordinates are expressed as WGS84 longitudes and latitudes (as required by RFC 7946); FALSE if they are in some other (undefined) CRS
<i>m_coords</i>	TRUE if M-values will be exported as ordinary coordinates; FALSE for strict RFC 7946 conformance (no M-Values at all)
<i>indent</i>	TRUE if the GeoJSON file will be properly indented for enhanced human readability; FALSE if the GeoJSON file will be in a single monolithic line without blank spaces.
<i>colname_case</i>	one between GAIA_DBF_COLNAME_LOWERCASE, GAIA_DBF_COLNAME_UPPERCASE or GAIA_DBF_COLNAME_CASE_IGNORE.
<i>rows</i>	on completion will contain the total number of exported rows
<i>error_message</i>	will point to a diagnostic error message in case of failure, otherwise NULL

See also

[dump_geojson](#), [dump_geojson_ex](#)

Returns

0 on failure, any other value on success

Note

you are expected to free before or later an eventual error message by calling `sqlite3_free()`

6.1.2.12 dump_geojson_ex()

```
SPATIALITE_DECLARE int dump_geojson_ex (  
    sqlite3 * sqlite,  
    char * table,  
    char * geom_col,  
    char * outfile_path,  
    int precision,  
    int option,  
    int * rows )
```

Dumps a full geometry-table into an external GeoJSON file (old specification)

Parameters

<i>sqlite</i>	handle to current DB connection
<i>table</i>	the name of the table to be exported
<i>geom_col</i>	the name of the geometry column
<i>outfile_path</i>	pathname for the GeoJSON file to be written to
<i>precision</i>	number of decimal digits for coordinates
<i>option</i>	the format to use for output
<i>rows</i>	on completion will contain the total number of exported rows

See also

[dump_geojson](#), [dump_geojson2](#)

Note

valid values for option are:

- 0 no option
- 1 GeoJSON MBR
- 2 GeoJSON Short CRS (e.g EPSG:4326)
- 3 MBR + Short CRS
- 4 GeoJSON Long CRS (e.g urn:ogc:def:crs:EPSG::4326)
- 5 MBR + Long CRS

Returns

0 on failure, any other value on success

6.1.2.13 dump_kml()

```
SPATIALITE_DECLARE int dump_kml (  
    sqlite3 * sqlite,  
    char * table,  
    char * geom_col,  
    char * kml_path,  
    char * name_col,  
    char * desc_col,  
    int precision )
```

Dumps a full geometry-table into an external KML file.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>table</i>	the name of the table to be exported
<i>geom_col</i>	the name of the geometry column
<i>kml_path</i>	pathname of the KML file to be exported
<i>name_col</i>	column to be used for KML "name" (may be null)
<i>desc_col</i>	column to be used for KML "description" (may be null)
<i>precision</i>	number of decimal digits for coordinates

See also

[dump_kml_ex](#)

Returns

0 on failure, any other value on success

6.1.2.14 dump_kml_ex()

```
SPATIALITE_DECLARE int dump_kml_ex (  
    sqlite3 * sqlite,  
    char * table,  
    char * geom_col,  
    char * kml_path,  
    char * name_col,  
    char * desc_col,  
    int precision,  
    int * rows )
```

Dumps a full geometry-table into an external KML file.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>table</i>	the name of the table to be exported
<i>geom_col</i>	the name of the geometry column
<i>kml_path</i>	pathname of the KML file to be exported
<i>name_col</i>	column to be used for KML "name" (may be null)
<i>desc_col</i>	column to be used for KML "description" (may be null)
<i>precision</i>	number of decimal digits for coordinates
<i>rows</i>	on completion will contain the total number of exported rows

See also

[dump_kml](#)

Returns

0 on failure, any other value on success

6.1.2.15 dump_shapefile()

```
SPATIALITE_DECLARE int dump_shapefile (
    sqlite3 * sqlite,
    char * table,
    char * column,
    char * shp_path,
    char * charset,
    char * geom_type,
    int verbose,
    int * rows,
    char * err_msg )
```

Dumps a full geometry-table into an external Shapefile.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>table</i>	the name of the table to be exported
<i>column</i>	the name of the geometry column
<i>shp_path</i>	pathname of the Shapefile to be exported (no suffix)
<i>charset</i>	a valid GNU ICONV charset to be used for DBF text strings
<i>geom_type</i>	"POINT", "LINESTRING", "POLYGON", "MULTIPOINT" or NULL
<i>verbose</i>	if TRUE a short report is shown on stderr
<i>rows</i>	on completion will contain the total number of exported rows
<i>err_msg</i>	on completion will contain an error message (if any)

See also

[dump_shapefile_ex](#)

Returns

0 on failure, any other value on success

6.1.2.16 dump_shapefile_ex()

```
SPATIALITE_DECLARE int dump_shapefile_ex (
    sqlite3 * sqlite,
    char * table,
    char * column,
    char * shp_path,
    char * charset,
    char * geom_type,
    int verbose,
    int * rows,
    int colcase_name,
    char * err_msg )
```

Dumps a full geometry-table into an external Shapefile.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>table</i>	the name of the table to be exported
<i>column</i>	the name of the geometry column
<i>shp_path</i>	pathname of the Shapefile to be exported (no suffix)
<i>charset</i>	a valid GNU ICONV charset to be used for DBF text strings
<i>geom_type</i>	"POINT", "LINESTRING", "POLYGON", "MULTIPOINT" or NULL
<i>verbose</i>	if TRUE a short report is shown on stderr
<i>rows</i>	on completion will contain the total number of exported rows
<i>colname_case</i>	one between GAIA_DBF_COLNAME_LOWERCASE, GAIA_DBF_COLNAME_UPPERCASE or GAIA_DBF_COLNAME_CASE_IGNORE.
<i>err_msg</i>	on completion will contain an error message (if any)

See also

[dump_shapefile](#)

Returns

0 on failure, any other value on success

6.1.2.17 dump_shapefile_ex2()

```
SPATIALITE_DECLARE int dump_shapefile_ex2 (
    sqlite3 * sqlite,
    void * proj_ctx,
    char * table,
    char * column,
    char * shp_path,
    char * charset,
    char * geom_type,
    int verbose,
    int * rows,
    int colcase_name,
    char * err_msg )
```

Dumps a full geometry-table into an external Shapefile.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>proj_ctx</i>	pointer to the current PROJ.6 context (may be NULL)
<i>table</i>	the name of the table to be exported
<i>column</i>	the name of the geometry column
<i>shp_path</i>	pathname of the Shapefile to be exported (no suffix)
<i>charset</i>	a valid GNU ICONV charset to be used for DBF text strings
<i>geom_type</i>	"POINT", "LINESTRING", "POLYGON", "MULTIPOINT" or NULL
<i>verbose</i>	if TRUE a short report is shown on stderr
<i>rows</i>	on completion will contain the total number of exported rows
<i>colname_case</i>	one between GAIA_DBF_COLNAME_LOWERCASE, GAIA_DBF_COLNAME_UPPERCASE or GAIA_DBF_COLNAME_CASE_IGNORE.
<i>err_msg</i>	on completion will contain an error message (if any)

See also

[dump_shapefile](#)

Returns

0 on failure, any other value on success

6.1.2.18 elementary_geometries()

```
SPATIALITE_DECLARE void elementary_geometries (
    sqlite3 * sqlite,
    char * inTable,
    char * geometry,
    char * outTable,
    char * pKey,
    char * multiId )
```

Creates a derived table surely containing elementary Geometries.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>inTable</i>	name of the input table
<i>geometry</i>	name of the Geometry column
<i>outTable</i>	name of the output table to be created
<i>pKey</i>	name of the Primary Key column in the output table
<i>multild</i>	name of the column identifying origins in the output table

See also

[elementary_geometries_ex](#)

Note

if the input table contains some kind of complex Geometry (MULTIPOINT, MULTILINESTRING, MULTIPOLYGON or GEOMETRYCOLLECTION), then many rows are inserted into the output table: each single row will contain the same attributes and an elementary Geometry. All the rows created by expanding the same input row will expose the same value in the "multild" column.

6.1.2.19 elementary_geometries_ex()

```
SPATIALITE_DECLARE void elementary_geometries_ex (
    sqlite3 * sqlite,
    char * inTable,
    char * geometry,
    char * outTable,
    char * pKey,
    char * multiId,
    int * rows )
```

Creates a derived table surely containing elementary Geometries.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>inTable</i>	name of the input table
<i>geometry</i>	name of the Geometry column
<i>outTable</i>	name of the output table to be created
<i>pKey</i>	name of the Primary Key column in the output table
<i>multild</i>	name of the column identifying origins in the output table
<i>rows</i>	on completion will contain the total number of inserted rows

See also

[elementary_geometries_ex2](#)

Note

if the input table contains some kind of complex Geometry (MULTIPOINT, MULTILINESTRING, MULTIPOLYGON or GEOMETRYCOLLECTION), then many rows are inserted into the output table: each single row will contain the same attributes and an elementary Geometry. All the rows created by expanding the same input row will expose the same value in the "multid" column.

6.1.2.20 elementary_geometries_ex2()

```
SPATIALITE_DECLARE void elementary_geometries_ex2 (
    sqlite3 * sqlite,
    char * inTable,
    char * geometry,
    char * outTable,
    char * pKey,
    char * multiId,
    int * rows,
    int transaction )
```

Creates a derived table surely containing elementary Geometries.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>inTable</i>	name of the input table
<i>geometry</i>	name of the Geometry column
<i>outTable</i>	name of the output table to be created
<i>pKey</i>	name of the Primary Key column in the output table
<i>multid</i>	name of the column identifying origins in the output table
<i>rows</i>	on completion will contain the total number of inserted rows
<i>transaction</i>	boolean; if set to TRUE will internally handle a SQL Transaction

See also

[elementary_geometries_ex3](#)

Note

if the input table contains some kind of complex Geometry (MULTIPOINT, MULTILINESTRING, MULTIPOLYGON or GEOMETRYCOLLECTION), then many rows are inserted into the output table: each single row will contain the same attributes and an elementary Geometry. All the rows created by expanding the same input row will expose the same value in the "multid" column.

6.1.2.21 elementary_geometries_ex3()

```
SPATIALITE_DECLARE void elementary_geometries_ex3 (
    sqlite3 * sqlite,
```

```

char * inTable,
char * geometry,
char * outTable,
char * pKey,
char * multiId,
const void * options,
int * rows,
int transaction )

```

Creates a derived table surely containing elementary Geometries.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>inTable</i>	name of the input table
<i>geometry</i>	name of the Geometry column
<i>outTable</i>	name of the output table to be created
<i>pKey</i>	name of the Primary Key column in the output table
<i>multild</i>	name of the column identifying origins in the output table
<i>options</i>	pointer to an Options list created by <code>gaiaAuxClonerCreate()</code>
<i>rows</i>	on completion will contain the total number of inserted rows
<i>transaction</i>	boolean; if set to TRUE will internally handle a SQL Transaction

See also

[elementary_geometries](#)

Note

if the input table contains some kind of complex Geometry (MULTIPOINT, MULTILINESTRING, MULTIPOLYGON or GEOMETRYCOLLECTION), then many rows are inserted into the output table: each single row will contain the same attributes and an elementary Geometry. All the rows created by expanding the same input row will expose the same value in the "multild" column.

6.1.2.22 enable_tiny_point()

```

SPATIALITE_DECLARE void enable_tiny_point (
    const void * ptr )

```

Enables the BLOB-TinyPoint encoding.

Parameters

<i>ptr</i>	the same memory pointer passed to the corresponding call to spatialite_init_ex() and returned by spatialite_alloc_connection()
------------	--

See also

[disable_tiny_point](#), [is_tiny_point_enabled](#)

6.1.2.23 gaia_create_routing()

```
SPATIALITE_DECLARE int gaia_create_routing (
    sqlite3 * db_handle,
    const void * cache,
    const char * routing_data_table,
    const char * virtual_routing_table,
    const char * input_table,
    const char * from_column,
    const char * to_column,
    const char * geom_column,
    const char * cost_column,
    const char * name_column,
    int a_star_enabled,
    int bidirectional,
    const char * oneway_from,
    const char * oneway_to,
    int overwrite )
```

Will attempt to create a VirtualRouting from an input table.

Parameters

<i>db_handle</i>	handle to the current SQLite connection
<i>cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>routing_data_table</i>	name of the Routing Data Table to be created.
<i>virtual_routing_table</i>	name of the VirtualRouting Table to be created.
<i>input_table</i>	name of the input table to be processed.
<i>from_column</i>	name of the input table column containing NodeFrom.
<i>to_column</i>	name of the input table column containing NodeTo.
<i>geom_column</i>	name of the input table column containing Linestring Geometries (could be eventually NULL).
<i>cost_column</i>	name of the input table column containing Cost values (could be eventually NULL).
<i>name_column</i>	name of the input table column containing RoadName (could be eventually NULL).
<i>a_star_enabled</i>	if set to TRUE the Routing Data Table will support both Djiskra's Shortest Path and A* algorithms; if set to FALSE only the Djiskra's algorithm will be supported.
<i>bidirectional</i>	if set to TRUE all input arcs/links will be assumed to be bidirectional (from-to and to-from); if set to FALSE all input arcs/links will be assumed to be unidirectional (from-to only).
<i>oneway_from</i>	name of the input table column containing OneWayFrom (could be eventually NULL).
<i>oneway_to</i>	name of the input table column containing OneWayTo (could be eventually NULL).
<i>overwrite</i>	if set to TRUE both the Routing Data Table and the VirtualRouting Table will be dropped if already existing; if set to FALSE an already existing Routing Data Table or VirtualRouting Table will cause a fatal error.

Returns

0 on failure, any other value on success

Note

at least one between `geom_column` and `cost_column` shall not be NULL. both `oneway_from` and `oneway_to` must be NULL or must contain a valid column name; mixing a column name and a NULL will be considered a fatal error.

6.1.2.24 gaia_create_routing_nodes()

```
SPATIALITE_DECLARE int gaia_create_routing_nodes (
    sqlite3 * db_handle,
    const void * cache,
    const char * prefix,
    const char * table,
    const char * geom_column,
    const char * from_column,
    const char * to_column )
```

Will attempt to create the Routing Nodes columns for a spatial table.

Parameters

<i>db_handle</i>	handle to the current SQLite connection
<i>cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>prefix</i>	schema prefix identifying the target DB (could be eventually NULL)
<i>table</i>	name of the table to be processed.
<i>geom_column</i>	name of the table column containing Linestring Geometries (could be eventually NULL).
<i>from_column</i>	name of the table column where NodeFrom values will be stored.
<i>to_column</i>	name of the column where ToFrom values will be stored.

Returns

0 on failure, any other value on success

6.1.2.25 gaiaCreateMetaCatalogTables()

```
SPATIALITE_DECLARE int gaiaCreateMetaCatalogTables (
    sqlite3 * handle )
```

Creates (or re-creates) the "splite_metacatalog" and "splite_metacatalog_statistics" tables.

Parameters

<i>handle</i>	SQLite handle to current DB connection.
---------------	---

Returns

0 (FALSE) on failure, any other value (TRUE) on success

See also

[gaiaUpdateMetaCatalogStatistics](#), [gaiaUpdateMetaCatalogStatisticsFromMaster](#)

6.1.2.26 **gaiaCutter()**

```
SPATIALITE_DECLARE int gaiaCutter (
    sqlite3 * db_handle,
    const void * cache,
    const char * in_db_prefix,
    const char * input_table,
    const char * input_geom,
    const char * blade_db_prefix,
    const char * blade_table,
    const char * blade_geom,
    const char * output_table,
    int transaction,
    int ram_tmp_store,
    char ** message )
```

Will precisely cut the input dataset against polygonal blade(s) and will consequently create and populate an output dataset.

Parameters

<i>db_handle</i>	handle to the current SQLite connection
<i>cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>in_db_prefix</i>	prefix of the database where the input table is expected to be found. if NULL then "MAIN" will be assumed.
<i>input_table</i>	name of the input table to be processed.
<i>input_geometry</i>	name of the input table Geometry column; it could be NULL and in this case the appropriate column name will be automatically determined. anyway if the input table do contains two or more Geometries passing a NULL geometry name will raise a fatal error.
<i>blade_db_prefix</i>	prefix of the database where the "blade" table is expected to be found. if NULL then "MAIN" will be assumed.
<i>blade_table</i>	name of the table expected to contain Polygons or MultiPolygon Geometries acting as blades.
<i>blade_geometry</i>	name of the "blade" table Geometry column; it could be NULL and in this case the appropriate column name will be automatically determined. anyway if the input table do contains two or more Geometries passing a NULL geometry name will raise a fatal error.
<i>output_table</i>	name to assigned to the destination table intended to permanently store all results. this table must non exists.
<i>transaction</i>	boolean; if set to TRUE will internally handle a SQL Transaction.
<i>ram_tmp_store</i>	boolean: if set to TRUE all TEMPORARY tables and indices will be created in RAM, otherwise in a file.
<i>message</i>	pointer to a string buffer; if not NULL it will point on completion an eventual error message.

Returns

0 on failure, any other value on success

Note

the message buffer if not NULL will point to a dynamic memory allocation and is expected to be released by calling `sqlite3_free()`

6.1.2.27 gaiaDropTable()

```
SPATIALITE_DECLARE int gaiaDropTable (
    sqlite3 * sqlite,
    const char * table )
```

Drops a layer-table, removing any related dependency.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>table</i>	name of the table to be removed

Note

this function will drop a SpatialTable, SpatialView or VirtualShape being properly registered within the Meta-data tables.
an eventual Spatial Index will be dropped as well, and any row referring the selected table will be removed from the Metadata tables.

Returns

0 on failure, any other value on success

See also

[gaiaDropTableEx](#), [gaiaRenameTable](#), [gaiaRenameColumn](#)

Note

this one simply is a convenience method always defaulting to `gaiaDropTableEx(sqlite, "main", table);`

6.1.2.28 gaiaDropTable5()

```
SPATIALITE_DECLARE int gaiaDropTable5 (
    sqlite3 * sqlite,
    const char * prefix,
    const char * table,
    char ** error_message )
```

Drops a layer-table, removing any related dependency.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>prefix</i>	schema prefix identifying the target DB "main" always identifies the main DB (primary, not Attached).
<i>table</i>	name of the table or view to be dropped
<i>error_message</i>	will point to a diagnostic error message in case of failure, otherwise NULL

Note

this function will drop a SpatialTable, SpatialView or VirtualShape being properly registered within the Meta-data tables.

an eventual Spatial Index will be dropped as well, and any row referring the selected table will be removed from the Metadata tables.

an eventual diagnostic message pointed by *error_message* must be freed by calling `sqlite3_free()`

Returns

0 on failure, any other value on success

See also

[gaiaDropTableEx3](#), [gaiaRenameTable](#), [gaiaRenameColumn](#)

6.1.2.29 gaiaDropTableEx()

```
SPATIALITE_DECLARE int gaiaDropTableEx (
    sqlite3 * sqlite,
    const char * prefix,
    const char * table )
```

Drops a layer-table, removing any related dependency.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>prefix</i>	schema prefix identifying the target DB "main" always identifies the main DB (primary, not Attached).
<i>table</i>	name of the table to be removed

Note

this function will drop a SpatialTable, SpatialView or VirtualShape being properly registered within the Meta-data tables.

an eventual Spatial Index will be dropped as well, and any row referring the selected table will be removed from the Metadata tables.

Returns

0 on failure, any other value on success

See also

[gaiaDropTableEx2](#), [gaiaRenameTable](#), [gaiaRenameColumn](#)

6.1.2.30 [gaiaDropTableEx2\(\)](#)

```
SPATIALITE_DECLARE int gaiaDropTableEx2 (  
    sqlite3 * sqlite,  
    const char * prefix,  
    const char * table,  
    int transaction )
```

Drops a layer-table, removing any related dependency.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>prefix</i>	schema prefix identifying the target DB "main" always identifies the main DB (primary, not Attached).
<i>table</i>	name of the table to be removed
<i>transaction</i>	boolean; if set to TRUE will internally handle a SQL Transaction

Note

this function will drop a SpatialTable, SpatialView or VirtualShape being properly registered within the Meta-data tables.

an eventual Spatial Index will be dropped as well, and any row referring the selected table will be removed from the Metadata tables.

Returns

0 on failure, any other value on success

See also

[gaiaDropTable](#), [gaiaRenameTable](#), [gaiaRenameColumn](#)

6.1.2.31 [gaiaDropTableEx3\(\)](#)

```
SPATIALITE_DECLARE int gaiaDropTableEx3 (  
    sqlite3 * sqlite,  
    const char * prefix,  
    const char * table,  
    int transaction,  
    char ** error_message )
```

Drops a layer-table, removing any related dependency.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>prefix</i>	schema prefix identifying the target DB "main" always identifies the main DB (primary, not Attached).
<i>table</i>	name of the table to be removed
<i>transaction</i>	boolean; if set to TRUE will internally handle a SQL Transaction
<i>error_message</i>	will point to a diagnostic error message in case of failure, otherwise NULL

Note

this function will drop a SpatialTable, SpatialView or VirtualShape being properly registered within the Meta-data tables.
 an eventual Spatial Index will be dropped as well, and any row referring the selected table will be removed from the Metadata tables.
 an eventual diagnostic message pointed by *error_message* must be freed by calling `sqlite3_free()`

Returns

0 on failure, any other value on success

See also

[gaiaDropTable5](#), [gaiaRenameTable](#), [gaiaRenameColumn](#)

Deprecated use [gaiaDropTable5\(\)](#) as a full replacement

6.1.2.32 `gaiaFreeVectorLayersList()`

```
SPATIALITE_DECLARE void gaiaFreeVectorLayersList (
    gaiaVectorLayersListPtr ptr )
```

Destroys a VectorLayersList object.

Parameters

<i>ptr</i>	pointer to the VectorLayersList object to be destroyed
------------	--

See also

[gaiaGetVectorLayersList](#)

Examples

[demo5.c](#).

6.1.2.33 `gaiaGetLayerExtent()`

```
SPATIALITE_DECLARE gaiaGeomCollPtr gaiaGetLayerExtent (
    sqlite3 * handle,
    const char * table,
    const char * geometry,
    int mode )
```

Queries the Metadata tables returning the Layer Full Extent.

Parameters

<i>handle</i>	SQLite handle to current DB connection.
<i>table</i>	VectorLayer Table (or View, or VirtualShape).
<i>geometry</i>	Geometry Column name.
<i>mode</i>	if TRUE a PESSIMISTIC statistics update will be implied, otherwise OPTIMISTIC.

Returns

the pointer to the newly created Geometry (Envelope): NULL on failure

See also

`update_layer_statistic`, [gaiaStatisticsInvalidate](#), [gaiaGetVectorLayersList](#)

Note

you are responsible to destroy (before or after) any allocated Geometry returned by [gaiaGetLayerExtent\(\)](#).
 The geometry arg is optional when the table simply has a single Geometry Column, and can be NULL in this case.
 When the mode arg is set to FALSE (default) then the returned infos will be simply retrieved from the staticized statistic tables (faster, but could be inaccurate).
 If the mode arg is set to TRUE a preliminary attempt to update the statistic tables will be always performed (probably slower, but surely accurate).
 If the named Layer doesn't exist, or if it's completely empty (not containing any valid Geometry) NULL will be returned.

6.1.2.34 `gaiaGetVectorLayersList()`

```
SPATIALITE_DECLARE gaiaVectorLayersListPtr gaiaGetVectorLayersList (
    sqlite3 * handle,
    const char * table,
    const char * geometry,
    int mode )
```

Queries the Metadata tables supporting Vector Layers.

Parameters

<i>handle</i>	SQLite handle to current DB connection.
<i>table</i>	VectorLayer Table (or View, or VirtualShape).
<i>geometry</i>	Geometry Column name.
<i>mode</i>	one of GAIA_VECTORS_LIST_OPTIMISTIC or GAIA_VECTORS_LIST_PESSIMISTIC.

Returns

the pointer to the newly created VectorLayersList object: NULL on failure

See also

[gaiaFreeVectorLayersList](#), [update_layer_statistics](#), [gaiaStatisticsInvalidate](#), [gaiaGetLayerExtent](#), [gaiaGetVectorLayersList](#)

Note

you are responsible to destroy (before or after) any allocated VectorLayersList returned by [gaiaGetVectorLayersList\(\)](#).

If the table arg is NULL all VectorLayers defined within the DB will be reported (and in this case all Attribute↔Field Infos will be always suppressed); otherwise only a single Layer will be reported (if existing).

By defining the geometry arg (not NULL) you can further restrict the returned report.

When the mode arg is set to GAIA_VECTORS_LIST_OPTIMISTIC (default) then the returned infos will be simply retrieved from the staticized statistic tables (faster, but could be inaccurate).

If the mode arg is set to GAIA_VECTORS_LIST_PESSIMISTIC a preliminary attempt to update the statistic tables will be always performed (probably slower, but surely accurate).

Examples

[demo5.c](#).

6.1.2.35 gaiaRenameColumn()

```
SPATIALITE_DECLARE int gaiaRenameColumn (
    sqlite3 * sqlite,
    const char * prefix,
    const char * table,
    const char * old_name,
    const char * new_name,
    char ** error_message )
```

Renames a Table's Column.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>prefix</i>	schema prefix identifying the target DB "main" always identifies the main DB (primary, not Attached).
<i>table</i>	name of the table containing the column to be renamed (always expected to be in the MAIN database).
<i>old_name</i>	current name of the column to be renamed
<i>new_name</i>	new column name to be set
<i>error_message</i>	will point to a diagnostic error message in case of failure, otherwise NULL

Note

this function will corretly rename a Geometry Column being properly registered within the Metadata tables.
all triggers, Spatial Index and alike will be correctly recovered.
an eventual diagnostic message pointed by `error_message` must be freed by calling `sqlite3_free()`

Returns

0 on failure, any other value on success

See also

[gaiaDropTable](#), [gaiaRenameTable](#)

Note

SQLite 3.25 (or later) is stricly required.

6.1.2.36 gaiaRenameTable()

```
SPATIALITE_DECLARE int gaiaRenameTable (
    sqlite3 * sqlite,
    const char * prefix,
    const char * old_name,
    const char * new_name,
    char ** error_message )
```

Renames a Table.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>prefix</i>	schema prefix identifying the target DB "main" always identifies the main DB (primary, not Attached).
<i>old_name</i>	current name of the table to be renamed (always expected to be in the MAIN database).
<i>new_name</i>	new table name to be set
<i>error_message</i>	will point to a diagnostic error message in case of failure, otherwise NULL

Note

this function will corretly rename a SpatialTable being properly registered within the Metadata tables.
all triggers, Spatial Index and alike will be correctly recovered.
an eventual diagnostic message pointed by `error_message` must be freed by calling `sqlite3_free()`

Returns

0 on failure, any other value on success

See also

[gaiaDropTable](#), [gaiaRenameColumn](#)

Note

SQLite 3.25 (or later) is stricly required.

6.1.2.37 gaiaStatisticsInvalidate()

```
SPATIALITE_DECLARE int gaiaStatisticsInvalidate (
    sqlite3 * handle,
    const char * table,
    const char * geometry )
```

Immediately and unconditionally invalidates the already existing Statistics.

Parameters

<i>handle</i>	SQLite handle to current DB connection.
<i>table</i>	VectorLayer Table (or View, or VirtualShape).
<i>geometry</i>	Geometry Column name.

Returns

0 on success, any other value on success

See also

[update_layer_statistics](#), [gaiaGetLayerExtent](#), [gaiaGetVectorLayersList](#)

Note

if the table arg is NULL all Statistics for any VectorLayer defined within the DB will be invalidated; otherwise only a single Layer will be affectedd (if existing).
By defining the geometry arg (not NULL) you can further restrict your selection.

6.1.2.38 gaiaUpdateMetaCatalogStatistics()

```
SPATIALITE_DECLARE int gaiaUpdateMetaCatalogStatistics (
    sqlite3 * handle,
    const char * table,
    const char * column )
```

Updates the "splite_metacatalog_statistics" table.

Parameters

<i>handle</i>	SQLite handle to current DB connection.
<i>table</i>	name of the table to be processed.
<i>column</i>	name of the column to be processed.

Returns

0 (FALSE) on failure, any other value (TRUE) on success

See also

[gaiaCreateMetaCatalogTables](#), [gaiaUpdateMetaCatalogStatisticsFromMaster](#)

6.1.2.39 `gaiaUpdateMetaCatalogStatisticsFromMaster()`

```
SPATIALITE_DECLARE int gaiaUpdateMetaCatalogStatisticsFromMaster (
    sqlite3 * handle,
    const char * master_table,
    const char * table_name,
    const char * column_name )
```

Updates the "splite_metacatalog_statistics" table (using a Master Table).

Parameters

<i>handle</i>	SQLite handle to current DB connection.
<i>master_table</i>	name of the master-table controlling the whole process.
<i>table_name</i>	name of the column into the master-table containing table-names.
<i>column_name</i>	name of the column into the master-table containing column-names.

Returns

0 (FALSE) on failure, any other value (TRUE) on success

See also

[gaiaCreateMetaCatalogTables](#), [gaiaUpdateMetaCatalogStatistics](#)

6.1.2.40 `insert_epsg_srid()`

```
SPATIALITE_DECLARE int insert_epsg_srid (
    sqlite3 * sqlite,
    int srid )
```

Inserts some inlined EPSG definition into the "spatial_ref_sys" table.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>srid</i>	the SRID value uniquely identifying the required EPSG definition

Returns

0 on failure, any other value on success

6.1.2.41 is_kml_constant()

```
SPATIALITE_DECLARE int is_kml_constant (
    sqlite3 * sqlite,
    char * table,
    char * column )
```

Checks if a column is actually defined into the given table.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>table</i>	the table to be checked
<i>column</i>	the column to be checked

Returns

0 on success, any other value on success

Note

internally used to detect if some KML attribute defaults to a constant value

6.1.2.42 is_tiny_point_enabled()

```
SPATIALITE_DECLARE int is_tiny_point_enabled (
    const void * ptr )
```

Checks if the BLOB-TinyPoint encoding is enabled or not.

Parameters

<i>ptr</i>	the same memory pointer passed to the corresponding call to spatialite_init_ex() and returned by spatialite_alloc_connection()
------------	--

Returns

0 (FALSE) if not enabled, any other value (TRUE) if enabled

See also

[enable_tiny_point](#), [disable_tiny_point](#)

6.1.2.43 load_dbf()

```
SPATIALITE_DECLARE int load_dbf (
    sqlite3 * sqlite,
    char * dbf_path,
    char * table,
    char * charset,
    int verbose,
    int * rows,
    char * err_msg )
```

Loads an external DBF file into a newly created table.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>dbf_path</i>	pathname of the DBF file to be imported
<i>table</i>	the name of the table to be created
<i>charset</i>	a valid GNU ICONV charset to be used for DBF text strings
<i>verbose</i>	if TRUE a short report is shown on stderr
<i>rows</i>	on completion will contain the total number of actually exported rows
<i>err_msg</i>	on completion will contain an error message (if any)

See also

[load_dbf_ex](#), [load_dbf_ex2](#), [load_dbf_ex3](#)

Note

this function simply calls `load_dbf_ex` by passing an implicit `pk_column=NULL` argument

Returns

0 on failure, any other value on success

6.1.2.44 load_dbf_ex()

```
SPATIALITE_DECLARE int load_dbf_ex (
    sqlite3 * sqlite,
    char * dbf_path,
    char * table,
    char * pk_column,
    char * charset,
    int verbose,
    int * rows,
    char * err_msg )
```

Loads an external DBF file into a newly created table.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>dbf_path</i>	pathname of the DBF file to be imported
<i>table</i>	the name of the table to be created
<i>pk_column</i>	name of the Primary Key column; if NULL or mismatching then "PK_UID" will be assumed by default.
<i>charset</i>	a valid GNU ICONV charset to be used for DBF text strings
<i>verbose</i>	if TRUE a short report is shown on stderr
<i>rows</i>	on completion will contain the total number of actually exported rows
<i>err_msg</i>	on completion will contain an error message (if any)

See also

[load_dbf](#), [load_dbf_ex2](#), [load_dbf_ex3](#)

Returns

0 on failure, any other value on success

6.1.2.45 load_dbf_ex2()

```
SPATIALITE_DECLARE int load_dbf_ex2 (
    sqlite3 * sqlite,
    char * dbf_path,
    char * table,
    char * pk_column,
    char * charset,
    int verbose,
    int text_date,
    int * rows,
    char * err_msg )
```

Loads an external DBF file into a newly created table.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>dbf_path</i>	pathname of the DBF file to be imported
<i>table</i>	the name of the table to be created
<i>pk_column</i>	name of the Primary Key column; if NULL or mismatching then "PK_UID" will be assumed by default.
<i>charset</i>	a valid GNU ICONV charset to be used for DBF text strings
<i>verbose</i>	if TRUE a short report is shown on stderr
<i>text_dates</i>	is TRUE all DBF dates will be considered as TEXT
<i>rows</i>	on completion will contain the total number of imported rows
<i>err_msg</i>	on completion will contain an error message (if any)

See also

[load_dbf](#), [load_dbf_ex](#), [load_dbf_ex3](#)

Returns

0 on failure, any other value on success

6.1.2.46 load_dbf_ex3()

```
SPATIALITE_DECLARE int load_dbf_ex3 (
    sqlite3 * sqlite,
    const char * dbf_path,
    const char * table,
    const char * pk_column,
    const char * charset,
    int verbose,
    int text_date,
    int * rows,
    int colname_case,
    char * err_msg )
```

Loads an external DBF file into a newly created table.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>dbf_path</i>	pathname of the DBF file to be imported
<i>table</i>	the name of the table to be created
<i>pk_column</i>	name of the Primary Key column; if NULL or mismatching then "PK_UID" will be assumed by default.
<i>charset</i>	a valid GNU ICONV charset to be used for DBF text strings
<i>verbose</i>	if TRUE a short report is shown on stderr
<i>text_dates</i>	is TRUE all DBF dates will be considered as TEXT
<i>rows</i>	on completion will contain the total number of imported rows
<i>colname_case</i>	one between GAIA_DBF_COLNAME_LOWERCASE, GAIA_DBF_COLNAME_UPPERCASE or GAIA_DBF_COLNAME_CASE_IGNORE.
<i>err_msg</i>	on completion will contain an error message (if any)

See also

[load_dbf](#), [load_dbf_ex](#), [load_dbf_ex2](#), [load_zip_dbf](#)

Returns

0 on failure, any other value on success

6.1.2.47 load_geojson()

```
SPATIALITE_DECLARE int load_geojson (
    sqlite3 * sqlite,
    char * path,
    char * table,
    char * column,
    int spatial_index,
    int srid,
    int colname_case,
    int * rows,
    char ** error_message )
```

Loads an external GeoJSON file into a newly created table.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>path</i>	pathname of the GeoJSON file to be imported
<i>table</i>	the name of the table to be created
<i>column</i>	the name of the geometry column. If NULL the column will be silently named "geometry".
<i>spatial_index</i>	if TRUE an R*Tree Spatial Index will be created
<i>srid</i>	when positive, the SRID value to be assigned to all Geometries. If 0 or negative SRID=4326 (lon-lat WGS84) will be always assumed accordingly to RFC 7946.
<i>colname_case</i>	one between GAIA_DBF_COLNAME_LOWERCASE, GAIA_DBF_COLNAME_UPPERCASE or GAIA_DBF_COLNAME_CASE_IGNORE.
<i>rows</i>	on completion will contain the total number of imported rows
<i>error_message</i>	will point to a diagnostic error message in case of failure, otherwise NULL

Returns

0 on failure, any other value on success

Note

you are expected to free before or later an eventual error message by calling `sqlite3_free()`

6.1.2.48 load_shapefile()

```
SPATIALITE_DECLARE int load_shapefile (
    sqlite3 * sqlite,
    char * shp_path,
    char * table,
    char * charset,
    int srid,
    char * column,
    int coerce2d,
    int compressed,
    int verbose,
    int spatial_index,
    int * rows,
    char * err_msg )
```

Loads an external Shapefile into a newly created table.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>shp_path</i>	pathname of the Shapefile to be imported (no suffix)
<i>table</i>	the name of the table to be created
<i>charset</i>	a valid GNU ICONV charset to be used for DBF text strings
<i>srid</i>	the SRID to be set for Geometries
<i>column</i>	the name of the geometry column
<i>coerce2d</i>	if TRUE any Geometry will be casted to 2D [XY]
<i>compressed</i>	if TRUE compressed Geometries will be created
<i>verbose</i>	if TRUE a short report is shown on stderr
<i>spatial_index</i>	if TRUE an R*Tree Spatial Index will be created
<i>rows</i>	on completion will contain the total number of imported rows
<i>err_msg</i>	on completion will contain an error message (if any)

Returns

0 on failure, any other value on success

See also

[load_shapefile_ex](#), [load_shapefile_ex2](#), [load_shapefile_ex3](#)

Note

this function simply calls `load_shapefile_ex` by passing implicit `gtype="AUTO"` and `pk_column=NULL` arguments

6.1.2.49 load_shapefile_ex()

```
SPATIALITE_DECLARE int load_shapefile_ex (
    sqlite3 * sqlite,
    char * shp_path,
    char * table,
    char * charset,
    int srid,
    char * geo_column,
    char * gtype,
    char * pk_column,
    int coerce2d,
    int compressed,
    int verbose,
    int spatial_index,
    int * rows,
    char * err_msg )
```

Loads an external Shapefile into a newly created table.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>shp_path</i>	pathname of the Shapefile to be imported (no suffix)
<i>table</i>	the name of the table to be created
<i>charset</i>	a valid GNU ICONV charset to be used for DBF text strings
<i>srid</i>	the SRID to be set for Geometries
<i>geo_column</i>	the name of the geometry column
<i>gtype</i>	expected to be one of: "LINESTRING", "LINESTRINGZ", "LINESTRINGM", "LINESTRINGZM", "MULTILINESTRING", "MULTILINESTRINGZ", "MULTILINESTRINGM", "MULTILINESTRINGZM", "POLYGON", "POLYGONZ", "POLYGONM", "POLYGONZM", "MULTIPOLYGON", "MULTIPOLYGONZ", "MULTIPOLYGONM", "MULTIPOLYGONZM" or "AUTO".
<i>pk_column</i>	name of the Primary Key column; if NULL or mismatching then "PK_UID" will be assumed by default.
<i>coerce2d</i>	if TRUE any Geometry will be casted to 2D [XY]
<i>compressed</i>	if TRUE compressed Geometries will be created
<i>verbose</i>	if TRUE a short report is shown on stderr
<i>spatial_index</i>	if TRUE an R*Tree Spatial Index will be created
<i>rows</i>	on completion will contain the total number of imported rows
<i>err_msg</i>	on completion will contain an error message (if any)

Returns

0 on failure, any other value on success

See also

[load_shapefile](#), [load_shapefile_ex2](#), [load_shapefile_ex3](#)

Note

the Shapefile format doesn't supports any distinction between LINESTRINGs and MULTILINESTRINGs, or between POLYGONs and MULTIPOLYGONs; as does not allows to clearly distinguish if the M-measure is required.

So a first preliminary scan of the Shapefile is required in order to correctly identify the actual payload (gtype = "AUTO", default case).

By explicitly specifying some expected geometry type this first scan will be skipped at all thus introducing a noticeable performance gain.

Anyway, declaring a mismatching geometry type will surely cause a failure.

6.1.2.50 load_shapefile_ex2()

```
SPATIALITE_DECLARE int load_shapefile_ex2 (
    sqlite3 * sqlite,
    char * shp_path,
    char * table,
    char * charset,
    int srid,
    char * geo_column,
    char * gtype,
    char * pk_column,
    int coerce2d,
    int compressed,
    int verbose,
    int spatial_index,
    int text_date,
    int * rows,
    char * err_msg )
```

Loads an external Shapefile into a newly created table.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>shp_path</i>	pathname of the Shapefile to be imported (no suffix)
<i>table</i>	the name of the table to be created
<i>charset</i>	a valid GNU ICONV charset to be used for DBF text strings
<i>srid</i>	the SRID to be set for Geometries
<i>geo_column</i>	the name of the geometry column
<i>gtype</i>	expected to be one of: "LINESTRING", "LINESTRINGZ", "LINESTRINGM", "LINESTRINGZM", "MULTILINESTRING", "MULTILINESTRINGZ", "MULTILINESTRINGM", "MULTILINESTRINGZM", "POLYGON", "POLYGONZ", "POLYGONM", "POLYGONZM", "MULTIPOLYGON", "MULTIPOLYGONZ", "MULTIPOLYGONM", "MULTIPOLYGONZM" or "AUTO".
<i>pk_column</i>	name of the Primary Key column; if NULL or mismatching then "PK_UID" will be assumed by default.
<i>coerce2d</i>	if TRUE any Geometry will be casted to 2D [XY]
<i>compressed</i>	if TRUE compressed Geometries will be created
<i>verbose</i>	if TRUE a short report is shown on stderr
<i>spatial_index</i>	if TRUE an R*Tree Spatial Index will be created
<i>text_dates</i>	is TRUE all DBF dates will be considered as TEXT
<i>rows</i>	on completion will contain the total number of imported rows
<i>err_msg</i>	on completion will contain an error message (if any)

Returns

0 on failure, any other value on success

See also

[load_shapefile](#), [load_shapefile_ex](#), [load_shapefile_ex3](#)

Note

the Shapefile format doesn't supports any distinction between LINESTRINGs and MULTILINESTRINGs, or between POLYGONs and MULTIPOLYGONs; as does not allows to clearly distinguish if the M-measure is required.

So a first preliminary scan of the Shapefile is required in order to correctly identify the actual payload (gtype = "AUTO", default case).

By explicitly specifying some expected geometry type this first scan will be skipped at all thus introducing a noticeable performance gain.

Anyway, declaring a mismatching geometry type will surely cause a failure.

6.1.2.51 load_shapefile_ex3()

```
SPATIALITE_DECLARE int load_shapefile_ex3 (
    sqlite3 * sqlite,
    const char * shp_path,
    const char * table,
    const char * charset,
    int srid,
    const char * geo_column,
    const char * gtype,
    const char * pk_column,
    int coerce2d,
    int compressed,
    int verbose,
    int spatial_index,
    int text_date,
    int * rows,
    int colname_case,
    char * err_msg )
```

Loads an external Shapefile into a newly created table.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>shp_path</i>	pathname of the Shapefile to be imported (no suffix)
<i>table</i>	the name of the table to be created
<i>charset</i>	a valid GNU ICONV charset to be used for DBF text strings
<i>srid</i>	the SRID to be set for Geometries
<i>geo_column</i>	the name of the geometry column
<i>gtype</i>	expected to be one of: "LINESTRING", "LINESTRINGZ", "LINESTRINGM", "LINESTRINGZM", "MULTILINESTRING", "MULTILINESTRINGZ", "MULTILINESTRINGM", "MULTILINESTRINGZM", "POLYGON", "POLYGONZ", "POLYGONM", "POLYGONZM", "MULTIPOLYGON", "MULTIPOLYGONZ", "MULTIPOLYGONM", "MULTIPOLYGONZM" or "AUTO".

Parameters

<i>pk_column</i>	name of the Primary Key column; if NULL or mismatching then "PK_UID" will be assumed by default.
<i>coerce2d</i>	if TRUE any Geometry will be casted to 2D [XY]
<i>compressed</i>	if TRUE compressed Geometries will be created
<i>verbose</i>	if TRUE a short report is shown on stderr
<i>spatial_index</i>	if TRUE an R*Tree Spatial Index will be created
<i>text_dates</i>	is TRUE all DBF dates will be considered as TEXT
<i>rows</i>	on completion will contain the total number of imported rows
<i>colname_case</i>	one between GAIA_DBF_COLNAME_LOWERCASE, GAIA_DBF_COLNAME_UPPERCASE or GAIA_DBF_COLNAME_CASE_IGNORE.
<i>err_msg</i>	on completion will contain an error message (if any)

Returns

0 on failure, any other value on success

See also

[load_shapefile](#), [load_shapefile_ex](#), [load_shapefile_ex2](#), [load_zip_shapefile](#)

Note

the Shapefile format doesn't supports any distinction between LINESTRINGs and MULTILINESTRINGs, or between POLYGONs and MULTIPOLYGONs; as does not allows to clearly distinguish if the M-measure is required.

So a first preliminary scan of the Shapefile is required in order to correctly identify the actual payload (gtype = "AUTO", default case).

By explicitly specifying some expected geometry type this first scan will be skipped at all thus introducing a noticeable performance gain.

Anyway, declaring a mismatching geometry type will surely cause a failure.

6.1.2.52 load_XL()

```
SPATIALITE_DECLARE int load_XL (
    sqlite3 * sqlite,
    const char * path,
    const char * table,
    unsigned int worksheetIndex,
    int first_titles,
    unsigned int * rows,
    char * err_msg )
```

Loads an external spreadsheet (.xls) file into a newly created table.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>path</i>	pathname of the spreadsheet file to be imported
<i>table</i>	the name of the table to be created
<i>worksheetIndex</i>	the index identifying the worksheet to be imported
<i>first_titles</i>	if TRUE the first line is assumed to contain column names
<i>rows</i>	on completion will contain the total number of actually exported rows

Returns

0 on failure, any other value on success

6.1.2.53 load_zip_dbf()

```
SPATIALITE_DECLARE int load_zip_dbf (
    sqlite3 * sqlite,
    const char * zip_file,
    const char * dbf_path,
    const char * table,
    const char * pk_column,
    const char * charset,
    int verbose,
    int text_date,
    int * rows,
    int colname_case,
    char * err_msg )
```

Loads an external DBF file (from Zipfile) into a newly created table.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>zip_path</i>	pathname of the Zipfile
<i>shp_path</i>	pseudo-pathname of the Shapefile to be imported (no suffix)
<i>sqlite</i>	handle to current DB connection
<i>zip_path</i>	pathname of the Zipfile
<i>filenamepseudo-pathname</i>	of the DBF file to be imported (including the '.dbf' suffix)
<i>table</i>	the name of the table to be created
<i>pk_column</i>	name of the Primary Key column; if NULL or mismatching then "PK_UID" will be assumed by default.
<i>charset</i>	a valid GNU ICONV charset to be used for DBF text strings
<i>verbose</i>	if TRUE a short report is shown on stderr
<i>text_dates</i>	is TRUE all DBF dates will be considered as TEXT
<i>rows</i>	on completion will contain the total number of imported rows
<i>colname_case</i>	one between GAIA_DBF_COLNAME_LOWERCASE, GAIA_DBF_COLNAME_UPPERCASE or GAIA_DBF_COLNAME_CASE_IGNORE.
<i>err_msg</i>	on completion will contain an error message (if any)

See also

[load_dbf_ex3](#)

Returns

0 on failure, any other value on success

6.1.2.54 load_zip_shapefile()

```
SPATIALITE_DECLARE int load_zip_shapefile (
    sqlite3 * sqlite,
    const char * zip_path,
    const char * shp_path,
    const char * table,
    const char * charset,
    int srid,
    const char * geo_column,
    const char * gtype,
    const char * pk_column,
    int coerce2d,
    int compressed,
    int verbose,
    int spatial_index,
    int text_date,
    int * rows,
    int colname_case,
    char * err_msg )
```

Loads an external Shapefile (from Zipfile) into a newly created table.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>zip_path</i>	pathname of the Zipfile
<i>shp_path</i>	pseudo-pathname of the Shapefile to be imported (no suffix)
<i>table</i>	the name of the table to be created
<i>charset</i>	a valid GNU ICONV charset to be used for DBF text strings
<i>srid</i>	the SRID to be set for Geometries
<i>geo_column</i>	the name of the geometry column
<i>gtype</i>	expected to be one of: "LINESTRING", "LINESTRINGZ", "LINESTRINGM", "LINESTRINGZM", "MULTILINESTRING", "MULTILINESTRINGZ", "MULTILINESTRINGM", "MULTILINESTRINGZM", "POLYGON", "POLYGONZ", "POLYGONM", "POLYGONZM", "MULTIPOLYGON", "MULTIPOLYGONZ", "MULTIPOLYGONM", "MULTIPOLYGONZM" or "AUTO".
<i>pk_column</i>	name of the Primary Key column; if NULL or mismatching then "PK_UID" will be assumed by default.
<i>coerce2d</i>	if TRUE any Geometry will be casted to 2D [XY]
<i>compressed</i>	if TRUE compressed Geometries will be created
<i>verbose</i>	if TRUE a short report is shown on stderr
<i>spatial_index</i>	if TRUE an R*Tree Spatial Index will be created
<i>text_dates</i>	is TRUE all DBF dates will be considered as TEXT
<i>rows</i>	on completion will contain the total number of imported rows
<i>colname_case</i>	one between GAIA_DBF_COLNAME_LOWERCASE, GAIA_DBF_COLNAME_UPPERCASE or GAIA_DBF_COLNAME_CASE_IGNORE.
<i>err_msg</i>	on completion will contain an error message (if any)

Returns

0 on failure, any other value on success

See also

[load_shapefile_ex3](#)

Note

the Shapefile format doesn't supports any distinction between LINESTRINGS and MULTILINESTRINGS, or between POLYGONS and MULTIPOLYGONS; as does not allows to clearly distinguish if the M-measure is required.

So a first preliminary scan of the Shapefile is required in order to correctly identify the actual payload (gtype = "AUTO", default case).

By explicitly specifying some expected geometry type this first scan will be skipped at all thus introducing a noticeable performance gain.

Anyway, declaring a mismatching geometry type will surely cause a failure.

6.1.2.55 `math_llabs()`

```
SPATIALITE_DECLARE sqlite3_int64 math_llabs (  
    sqlite3_int64 value )
```

A portable replacement for C99 llabs()

Parameters

<i>value</i>	a 64 bit integer value
--------------	------------------------

Returns

the corresponding absolute value

6.1.2.56 `math_round()`

```
SPATIALITE_DECLARE double math_round (  
    double value )
```

A portable replacement for C99 round()

Parameters

<i>value</i>	a double value
--------------	----------------

Returns

the nearest integral value

6.1.2.57 remove_duplicated_rows()

```
SPATIALITE_DECLARE void remove_duplicated_rows (
    sqlite3 * sqlite,
    char * table )
```

Remove duplicated rows from a table.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>table</i>	name of the table to be cleaned

See also

[check_duplicated_rows](#), [remove_duplicated_rows_ex](#)

Note

when two (or more) duplicated rows exist, only the first occurrence will be preserved, then deleting any further occurrence.

6.1.2.58 remove_duplicated_rows_ex()

```
SPATIALITE_DECLARE void remove_duplicated_rows_ex (
    sqlite3 * sqlite,
    char * table,
    int * removed )
```

Remove duplicated rows from a table.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>table</i>	name of the table to be cleaned
<i>removed</i>	on successful completion will contain the total count of removed duplicate rows

See also

[check_duplicated_rows](#), [remove_duplicated_rows_ex2](#)

Note

when two (or more) duplicated rows exist, only the first occurrence will be preserved, then deleting any further occurrence.

6.1.2.59 remove_duplicated_rows_ex2()

```
SPATIALITE_DECLARE void remove_duplicated_rows_ex2 (
    sqlite3 * sqlite,
    char * table,
    int * removed,
    int transaction )
```

Remove duplicated rows from a table.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>table</i>	name of the table to be cleaned
<i>removed</i>	on successful completion will contain the total count of removed duplicate rows
<i>transaction</i>	boolean; if set to TRUE will internally handle a SQL Transaction

See also

[check_duplicated_rows](#), [remove_duplicated_rows](#)

Note

when two (or more) duplicated rows exist, only the first occurrence will be preserved, then deleting any further occurrence.

6.1.2.60 sanitize_all_geometry_columns()

```
SPATIALITE_DECLARE int sanitize_all_geometry_columns (
    sqlite3 * sqlite,
    const char * tmp_prefix,
    const char * output_dir,
    int * not_repaired,
    char ** err_msg )
```

Sanitizes all Geometry Columns making all invalid geometries to be valid.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>tmp_prefix</i>	name-prefix for temporary tables
<i>output_dir</i>	pathname of the directory to be created for report-files
<i>not_repaired</i>	if this variable is not NULL on successful completion will contain the total count of repair failures (i.e. Geometries beyond possible repair)
<i>err_msg</i>	if this variable is not NULL and the return status is ZERO (failure), an appropriate error message will be returned

See also

[sanitize_all_geometry_columns_r](#), [check_geometry_column](#), [check_all_geometry_columns](#), [sanitize_geometry_column](#)

Note

this function will attempt to make valid all invalid geometries found within all Geometry Columns (vector layers); a temporary table is required so to support each input table.
 if the process has full success the temporary table will be deleted; otherwise it will be preserved for further inspection. a HTML report will be produced as well.
 an eventual error message returned via `err_msg` requires to be deallocated by invoking `free()`
 not reentrant and thread unsafe.

Returns

0 on failure, any other value on success

6.1.2.61 `sanitize_all_geometry_columns_r()`

```
SPATIALITE_DECLARE int sanitize_all_geometry_columns_r (
    const void * p_cache,
    sqlite3 * sqlite,
    const char * tmp_prefix,
    const char * output_dir,
    int * not_repaired,
    char ** err_msg )
```

Sanitizes all Geometry Columns making all invalid geometries to be valid.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>sqlite</i>	handle to current DB connection
<i>tmp_prefix</i>	name-prefix for temporary tables
<i>output_dir</i>	pathname of the directory to be created for report-files
<i>not_repaired</i>	if this variable is not NULL on successful completion will contain the total count of repair failures (i.e. Geometries beyond possible repair)
<i>err_msg</i>	if this variable is not NULL and the return status is ZERO (failure), an appropriate error message will be returned

See also

[sanitize_all_geometry_columns](#), [check_geometry_column](#), [check_all_geometry_columns](#), [sanitize_geometry_column](#)

Note

this function will attempt to make valid all invalid geometries found within all Geometry Columns (vector layers); a temporary table is required so to support each input table.
 if the process has full success the temporary table will be deleted; otherwise it will be preserved for further

inspection. a HTML report will be produced as well.
 an eventual error message returned via `err_msg` requires to be deallocated by invoking `free()`
 reentrant and thread-safe.

Returns

0 on failure, any other value on success

6.1.2.62 `sanitize_geometry_column()`

```
SPATIALITE_DECLARE int sanitize_geometry_column (
    sqlite3 * sqlite,
    const char * table,
    const char * geom,
    const char * tmp_table,
    const char * report_path,
    int * n_invalids,
    int * n_repaired,
    int * n_discarded,
    int * n_failures,
    char ** err_msg )
```

Sanitizes a Geometry Column making all invalid geometries to be valid.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>table</i>	name of the table
<i>geometry</i>	name of the column to be checked
<i>tmp_table</i>	name of the temporary table
<i>report_path</i>	pathname of the report-file
<i>n_invalids</i>	if this variable is not NULL on successful completion will contain the total number of invalid Geometries found into the sanitize table
<i>n_repaired</i>	if this variable is not NULL on successful completion will contain the total number of repaired Geometries
<i>n_discarded</i>	if this variable is not NULL on successful completion will contain the total number of repaired Geometries (by discarding fragments)
<i>n_failures</i>	if this variable is not NULL on successful completion will contain the total number of repair failures (i.e. Geometries beyond possible repair)
<i>err_msg</i>	if this variable is not NULL and the return status is ZERO (failure), an appropriate error message will be returned

See also

[sanitize_geometry_column_r](#), [check_geometry_column](#), [check_all_geometry_columns](#), [sanitize_all_geometry_columns](#)

Note

this function will attempt to make valid all invalid geometries found within a Geometry Column (layer); a temporary table is required.

if the process has full success the temporary table will be deleted; otherwise it will be preserved for further inspection. a HTML report will be produced as well.

an eventual error message returned via `err_msg` requires to be deallocated by invoking `free()`

not reentrant and thread unsafe.

Returns

0 on failure, any other value on success

6.1.2.63 `sanitize_geometry_column_r()`

```
SPATIALITE_DECLARE int sanitize_geometry_column_r (
    const void * p_cache,
    sqlite3 * sqlite,
    const char * table,
    const char * geom,
    const char * tmp_table,
    const char * report_path,
    int * n_invalids,
    int * n_repaired,
    int * n_discarded,
    int * n_failures,
    char ** err_msg )
```

Sanitizes a Geometry Column making all invalid geometries to be valid.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>sqlite</i>	handle to current DB connection
<i>table</i>	name of the table
<i>geometry</i>	name of the column to be checked
<i>tmp_table</i>	name of the temporary table
<i>report_path</i>	pathname of the report-file
<i>n_invalids</i>	if this variable is not NULL on successful completion will contain the total number of invalid Geometries found into the sanitize table
<i>n_repaired</i>	if this variable is not NULL on successful completion will contain the total number of repaired Geometries
<i>n_discarded</i>	if this variable is not NULL on successful completion will contain the total number of repaired Geometries (by discarding fragments)
<i>n_failures</i>	if this variable is not NULL on successful completion will contain the total number of repair failures (i.e. Geometries beyond possible repair)
<i>err_msg</i>	if this variable is not NULL and the return status is ZERO (failure), an appropriate error message will be returned

See also

[sanitize_geometry_column](#), [check_geometry_column](#), [check_all_geometry_columns](#), [sanitize_all_geometry_columns](#)

Note

this function will attempt to make valid all invalid geometries found within a Geometry Column (layer); a temporary table is required.

if the process has full success the temporary table will be deleted; otherwise it will be preserved for further inspection. a HTML report will be produced as well.

an eventual error message returned via `err_msg` requires to be deallocated by invoking `free()`
reentrant and thread-safe.

Returns

0 on failure, any other value on success

6.1.2.64 spatial_ref_sys_init()

```
SPATIALITE_DECLARE int spatial_ref_sys_init (
    sqlite3 * sqlite,
    int verbose )
```

Inserts the inlined EPSG dataset into the "spatial_ref_sys" table.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>verbose</i>	if TRUE a short report is shown on stderr

Returns

0 on failure, any other value on success

See also

[spatial_ref_sys_init2](#)

Note

this function is internally invoked by the SQL function `InitSpatialMetadata()`, and is not usually intended for direct use. This function is now deprecated, and will simply call `spatial_ref_sys_init2(sqlite, GAIA_EPSG_↵ ANY, verbose)`.

6.1.2.65 spatial_ref_sys_init2()

```
SPATIALITE_DECLARE int spatial_ref_sys_init2 (
    sqlite3 * sqlite,
    int mode,
    int verbose )
```

Inserts the inlined EPSG dataset into the "spatial_ref_sys" table.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>mode</i>	can be one of GAIA_EPSG_ANY, GAIA_EPSG_NONE or GAIA_EPSG_WGS84_ONLY
<i>verbose</i>	if TRUE a short report is shown on stderr

Returns

0 on failure, any other value on success

Note

this function is internally invoked by the SQL function InitSpatialMetadata(), and is not usually intended for direct use.

6.1.2.66 spatialite_alloc_connection()

```
SPATIALITE_DECLARE void* spatialite_alloc_connection (
    void )
```

Initializes the internal memory block supporting each connection.

See also

[spatialite_init_ex](#), [spatialite_cleanup_ex](#)

Examples

[demo1.c](#), [demo2.c](#), [demo3.c](#), [demo4.c](#), and [demo5.c](#).

6.1.2.67 spatialite_cleanup()

```
SPATIALITE_DECLARE void spatialite_cleanup (
    void )
```

Cleanup a Spatialite connection.

This function is now **DEPRECATED**; use [spatialite_cleanup_ex\(\)](#) for all new development.

This function performs general cleanup, essentially undoing the effect of [spatialite_init\(\)](#).

See also

[spatialite_init](#)

6.1.2.68 `spatialite_cleanup_ex()`

```
SPATIALITE_DECLARE void spatialite_cleanup_ex (
    const void * ptr )
```

Cleanup a Spatialite connection.

This function performs general cleanup, essentially undoing the effect of [spatialite_init_ex\(\)](#).

Parameters

<i>ptr</i>	the same memory pointer passed to the corresponding call to spatialite_init_ex() and returned by spatialite_alloc_connection()
------------	--

See also

[spatialite_init_ex](#), [spatialite_alloc_connection](#)

Examples

[demo1.c](#), [demo2.c](#), [demo3.c](#), [demo4.c](#), and [demo5.c](#).

6.1.2.69 `spatialite_finalize_topologies()`

```
SPATIALITE_DECLARE void spatialite_finalize_topologies (
    const void * ptr )
```

Partially Cleaning-up a Spatialite connection.

This function will destroy all TopoGeo and TopoNet objects from within a local cache.

Parameters

<i>ptr</i>	the same memory pointer passed to the corresponding call to spatialite_init_ex() and returned by spatialite_alloc_connection()
------------	--

See also

[spatialite_init_ex](#), [spatialite_alloc_connection](#)

6.1.2.70 `spatialite_init()`

```
SPATIALITE_DECLARE void spatialite_init (
    int verbose )
```

Initializes a Spatialite connection.

This function is now **DEPRECATED** because is not reentrant (not thread safe); use [spatialite_init_ex\(\)](#) for all new development.

Parameters

<i>verbose</i>	if TRUE a short start-up message is shown on stderr
----------------	---

See also

[spatialite_cleanup](#), [spatialite_init_ex](#)

Note

You absolutely must invoke this function before attempting to perform any other Spatialite's call.

6.1.2.71 `spatialite_init_ex()`

```
SPATIALITE_DECLARE void spatialite_init_ex (  
    sqlite3 * db_handle,  
    const void * ptr,  
    int verbose )
```

Initializes a Spatialite connection.

Parameters

<i>db_handle</i>	handle to the current SQLite connection
<i>ptr</i>	a memory pointer returned by spatialite_alloc_connection()
<i>verbose</i>	if TRUE a short start-up message is shown on stderr

See also

[spatialite_alloc_connection](#), [spatialite_cleanup_ex](#), [spatialite_init](#)

Note

You absolutely must invoke this function before attempting to perform any other Spatialite's call.

Examples

[demo1.c](#), [demo2.c](#), [demo3.c](#), [demo4.c](#), and [demo5.c](#).

6.1.2.72 `spatialite_init_geos()`

```
SPATIALITE_DECLARE void spatialite_init_geos (  
    void )
```

Initializes the GEOS library.

Note

You are never supposed to invoke this function (internally handled).

6.1.2.73 `spatialite_initialize()`

```
SPATIALITE_DECLARE void spatialite_initialize (
    void )
```

Initializes the library.

Note

you are always expected to explicitly call this function before attempting to call any Spatialite own function.

6.1.2.74 `spatialite_set_silent_mode()`

```
SPATIALITE_DECLARE void spatialite_set_silent_mode (
    const void * ptr )
```

Disables reporting GEOS/RTTOPO error and warnings on stderr.

Parameters

<i>ptr</i>	a memory pointer returned by spatialite_alloc_connection()
------------	--

See also

[spatialite_init_ex](#), [spatialite_cleanup_ex](#), [spatialite_set_verbose_mode](#)

6.1.2.75 `spatialite_set_verbode_mode()`

```
SPATIALITE_DECLARE void spatialite_set_verbode_mode (
    const void * ptr )
```

Enables reporting GEOS/RTTOPO error and warnings on stderr.

Parameters

<i>ptr</i>	a memory pointer returned by spatialite_alloc_connection()
------------	--

See also

[spatialite_init_ex](#), [spatialite_cleanup_ex](#), [spatialite_set_silent_mode](#)

6.1.2.76 `spatialite_shutdown()`

```
SPATIALITE_DECLARE void spatialite_shutdown (
    void )
```

Finalizes the library.

Note

you are always expected to explicitly call this function immediately before exiting the main application. This function will free any memory allocation and will release any system resource internally used by the library.

Examples

[demo1.c](#), [demo2.c](#), [demo3.c](#), [demo4.c](#), and [demo5.c](#).

6.1.2.77 `spatialite_target_cpu()`

```
SPATIALITE_DECLARE const char* spatialite_target_cpu (
    void )
```

Return the target CPU name.

Returns

the target CPU string.

6.1.2.78 `spatialite_version()`

```
SPATIALITE_DECLARE const char* spatialite_version (
    void )
```

Return the current library version.

Returns

the version string.

Examples

[demo1.c](#), [demo3.c](#), [demo4.c](#), and [demo5.c](#).

6.1.2.79 `srid_get_axis()`

```
SPATIALITE_DECLARE char* srid_get_axis (
    sqlite3 * sqlite,
    int srid,
    char axis,
    char mode )
```

checks a SRID definition from the "spatial_ref_sys" table then returning an Axis definition

Parameters

<i>sqlite</i>	handle to current DB connection
<i>srid</i>	the SRID value uniquely identifying the required EPSG definition
<i>axis</i>	should be one of SPLITE_AXIS_1 or SPLITE_AXIS_2
<i>mode</i>	should be one of SPLITE_AXIS_NAME or SPLITE_AXIS_ORIENTATION

Returns

the requested name on succes, NULL on failure

Note

you are responsible for freeing the returned name.

6.1.2.80 srid_get_datum()

```
SPATIALITE_DECLARE char* srid_get_datum (  
    sqlite3 * sqlite,  
    int srid )
```

checks a SRID definition from the "spatial_ref_sys" table then returning the corresponding Datum name

Parameters

<i>sqlite</i>	handle to current DB connection
<i>srid</i>	the SRID value uniquely identifying the required EPSG definition

Returns

the Datum name on succes, NULL on failure

Note

you are responsible for freeing the returned name.

6.1.2.81 srid_get_prime_meridian()

```
SPATIALITE_DECLARE char* srid_get_prime_meridian (  
    sqlite3 * sqlite,  
    int srid )
```

checks a SRID definition from the "spatial_ref_sys" table then returning the corresponding Prime Meridian name

Parameters

<i>sqlite</i>	handle to current DB connection
<i>srid</i>	the SRID value uniquely identifying the required EPSG definition

Returns

the Prime Meridian name on succes, NULL on failure

Note

you are responsible for freeing the returned name.

6.1.2.82 srid_get_projection()

```
SPATIALITE_DECLARE char* srid_get_projection (  
    sqlite3 * sqlite,  
    int srid )
```

checks a SRID definition from the "spatial_ref_sys" table then returning the corresponding Projection name

Parameters

<i>sqlite</i>	handle to current DB connection
<i>srid</i>	the SRID value uniquely identifying the required EPSG definition

Returns

the Projection name on succes, NULL on failure

Note

you are responsible for freeing the returned name.

6.1.2.83 srid_get_spheroid()

```
SPATIALITE_DECLARE char* srid_get_spheroid (  
    sqlite3 * sqlite,  
    int srid )
```

checks a SRID definition from the "spatial_ref_sys" table then returning the corresponding Spheroid name

Parameters

<i>sqlite</i>	handle to current DB connection
<i>srid</i>	the SRID value uniquely identifying the required EPSG definition

Returns

the Spheroid name on succes, NULL on failure

Note

you are responsible for freeing the returned name.

6.1.2.84 srid_get_unit()

```
SPATIALITE_DECLARE char* srid_get_unit (  
    sqlite3 * sqlite,  
    int srid )
```

checks a SRID definition from the "spatial_ref_sys" table then returning the corresponding Unit name

Parameters

<i>sqlite</i>	handle to current DB connection
<i>srid</i>	the SRID value uniquely identifying the required EPSG definition

Returns

the Unit name on succes, NULL on failure

Note

you are responsible for freeing the returned name.

6.1.2.85 srid_has_flipped_axes()

```
SPATIALITE_DECLARE int srid_has_flipped_axes (  
    sqlite3 * sqlite,  
    int srid,  
    int * flipped )
```

checks a SRID definition from the "spatial_ref_sys" table determining if the axes order is X-Y or Y-X

Parameters

<i>sqlite</i>	handle to current DB connection
<i>srid</i>	the SRID value uniquely identifying the required EPSG definition
<i>flipped</i>	on successful completion will contain 0 (FALSE) if axes order is X-Y, any other value (TRUE) if axes order is Y-X.

Returns

0 on failure, any other value on success

6.1.2.86 srid_is_geographic()

```
SPATIALITE_DECLARE int srid_is_geographic (  
    sqlite3 * sqlite,  
    int srid,  
    int * geographic )
```

checks a SRID definition from the "spatial_ref_sys" table determining if it is of the geographic type

Parameters

<i>sqlite</i>	handle to current DB connection
<i>srid</i>	the SRID value uniquely identifying the required EPSG definition
<i>geographic</i>	on successful completion will contain TRUE or FALSE

Returns

0 on failure, any other value on success

6.1.2.87 srid_is_projected()

```
SPATIALITE_DECLARE int srid_is_projected (  
    sqlite3 * sqlite,  
    int srid,  
    int * projected )
```

checks a SRID definition from the "spatial_ref_sys" table determining if it is of the projected type

Parameters

<i>sqlite</i>	handle to current DB connection
<i>srid</i>	the SRID value uniquely identifying the required EPSG definition
<i>projected</i>	on successful completion will contain TRUE or FALSE

Returns

0 on failure, any other value on success

6.1.2.88 update_layer_statistics()

```
SPATIALITE_DECLARE int update_layer_statistics (
    sqlite3 * sqlite,
    const char * table,
    const char * column )
```

Updates the LAYER_STATICS metadata table.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>table</i>	name of the table to be processed
<i>column</i>	name of the geometry to be processed

Note

this function will explore the given table/geometry determining the number of rows and the full layer extent; a corresponding table/geometry entry is expected to be already declared in the GEOMETRY_COLUMNS table. These informations will be permanently stored into the LAYER_STATISTICS table; if such table does not yet exists will be implicitly created.

- if table is NULL, any entry found within GEOMETRY_COLUMNS will be processed.
- if table is not NULL and column is NULL, any geometry belonging to the given table will be processed.
- if both table and column are not NULL, then only the given entry will be processed.

See also

[gaiaStatisticsInvalidate](#), [gaiaGetLayerExtent](#), [gaiaGetVectorLayersList](#)

Returns

0 on failure, the total count of processed entries on success

6.2 src/headers/spatialite/gaiaaux.h File Reference

Auxiliary/helper functions.

Macros

- `#define GAIA_SQL_SINGLE_QUOTE 1001`
SQL single quoted string (text constant)
- `#define GAIA_SQL_DOUBLE_QUOTE 1002`
SQL double quoted string (SQL name)

Functions

- GAIAAUX_DECLARE const char * [gaiaGetLocaleCharset](#) (void)
Retrieves the Locale Charset.
- GAIAAUX_DECLARE int [gaiaConvertCharset](#) (char **buf, const char *fromCs, const char *toCs)
Converts a text string from one charset to another.
- GAIAAUX_DECLARE void * [gaiaCreateUTF8Converter](#) (const char *fromCS)
Creates a persistent UTF8 converter object.
- GAIAAUX_DECLARE void [gaiaFreeUTF8Converter](#) (void *cvtCS)
Destroys an UTF8 converter object.
- GAIAAUX_DECLARE char * [gaiaConvertToUTF8](#) (void *cvtCS, const char *buf, int len, int *err)
Converts a text string to UTF8.
- GAIAAUX_DECLARE int [gaialsReservedSqliteName](#) (const char *name)
Checks if a name is a reserved SQLite name.
- GAIAAUX_DECLARE int [gaialsReservedSqlName](#) (const char *name)
Checks if a name is a reserved SQL name.
- GAIAAUX_DECLARE int [gaialllegalSqlName](#) (const char *name)
Checks if a name is an illegal SQL name.
- GAIAAUX_DECLARE char * [gaiaSingleQuotedSql](#) (const char *value)
Properly formats an SQL text constant.
- GAIAAUX_DECLARE char * [gaiaDoubleQuotedSql](#) (const char *value)
Properly formats an SQL name.
- GAIAAUX_DECLARE char * [gaiaQuotedSql](#) (const char *value, int quote)
Properly formats an SQL generic string.
- GAIAAUX_DECLARE char * [gaiaDequotedSql](#) (const char *value)
Properly formats an SQL generic string (dequoting)
- GAIAAUX_DECLARE void [gaiaCleanSqlString](#) (char *value)
deprecated function
- GAIAAUX_DECLARE void [gaiaInsertIntoSqlLog](#) (sqlite3 *sqlite, const char *user_agent, const char *utf8Sql, sqlite3_int64 *sqllog_pk)
SQL log: statement start.
- GAIAAUX_DECLARE void [gaiaUpdateSqlLog](#) (sqlite3 *sqlite, sqlite3_int64 sqllog_pk, int success, const char *errMsg)
SQL log: statement start.
- GAIAAUX_DECLARE void * [gaiaCreateMD5Checksum](#) (void)
Creates a persistent MD5 checksum object.
- GAIAAUX_DECLARE void [gaiaFreeMD5Checksum](#) (void *md5)
Destroys an MD5 checksum object.
- GAIAAUX_DECLARE void [gaiaUpdateMD5Checksum](#) (void *md5, const unsigned char *blob, int blob_len)
Updates an MD5 checksum object.
- GAIAAUX_DECLARE char * [gaiaFinalizeMD5Checksum](#) (void *md5)
Return an MD5 checksum value.
- GAIAAUX_DECLARE int [gaiaParseDMS](#) (const char *dms, double *longitude, double *latitude)
Return longitude and latitude angles from a DMS string.
- GAIAAUX_DECLARE char * [gaiaConvertToDMSEX](#) (double longitude, double latitude, int decimal_digits)
Return a DMS string - extended.
- GAIAAUX_DECLARE char * [gaiaConvertToDMS](#) (double longitude, double latitude)
Return a DMS string.
- GAIAAUX_DECLARE char * [gaiaEncodeURL](#) (const char *url, const char *in_charset)
Return a percent-encoded URL.
- GAIAAUX_DECLARE char * [gaiaDecodeURL](#) (const char *encoded, const char *out_charset)

- Return a clean URL from its percent-encoded representation.*

• GAIAAUX_DECLARE char * [gaiaDirNameFromPath](#) (const char *path)

Return the DirName component (if any) from a Path.
- GAIAAUX_DECLARE char * [gaiaFullFileNameFromPath](#) (const char *path)

Return the FullFileName from a Path.
- GAIAAUX_DECLARE char * [gaiaFileNameFromPath](#) (const char *path)

Return the FileName from a Path.
- GAIAAUX_DECLARE char * [gaiaFileExtFromPath](#) (const char *path)

Return the FileExtension from a Path.
- GAIAAUX_DECLARE char * [gaiaRemoveExtraSpaces](#) (const char *string)

Return a text string containing no repeated whitespaces.

6.2.1 Detailed Description

Auxiliary/helper functions.

6.2.2 Function Documentation

6.2.2.1 [gaiaCleanSqlString\(\)](#)

```
GAIAAUX_DECLARE void gaiaCleanSqlString (
    char * value )
```

deprecated function

Parameters

<i>value</i>	the string to be formatted
--------------	----------------------------

See also

[gaiaQuotedSql](#)

Note

this function is still supported simply for backward compatibility. it's intrinsically unsafe (passing huge strings potentially leads to buffer overflows) and you are strongly encouraged to use [gaiaQuotedSql\(\)](#) as a safest replacement.

6.2.2.2 [gaiaConvertCharset\(\)](#)

```
GAIAAUX_DECLARE int gaiaConvertCharset (
    char ** buf,
```



```
const char * fromCs,  
const char * toCs )
```

Converts a text string from one charset to another.

Parameters

<i>buf</i>	the text string to be converted
<i>fromCs</i>	the GNU ICONV name identifying the input charset
<i>toCs</i>	the GNU ICONV name identifying the output charset

Returns

0 on failure, any other value on success.

Note

this function uses an internal buffer limited to 64KB; so it's not safe passing extremely huge-sized text string.

6.2.2.3 gaiaConvertToDMS()

```
GAIAAUX_DECLARE char* gaiaConvertToDMS (
    double longitude,
    double latitude )
```

Return a DMS string.

Parameters

<i>longitude</i>	the angle of longitude expressed in Decimal Degrees.
<i>latitude</i>	the angle of latitude expressed in Decimal Degrees.

Returns

the corresponding DMS (Degrees/Minutes/Seconds) text string, or NULL on failure

See also

[gaiaConvertToDMSex](#)

Note

this function will return a dynamically allocated buffer created by malloc(). You are required to explicitly free() any string returned by this function.

this functions simply defaults to gaiaConvertToDMSex(lon, lat, 0)

6.2.2.4 gaiaConvertToDMSex()

```
GAIAAUX_DECLARE char* gaiaConvertToDMSex (
    double longitude,
    double latitude,
    int decimal_digits )
```

Return a DMS string - extended.

Parameters

<i>longitude</i>	the angle of longitude expressed in Decimal Degrees.
<i>latitude</i>	the angle of latitude expressed in Decimal Degrees.
<i>decimal_digits</i>	how many decimal digits must be preserved for representing Seconds.

Returns

the corresponding DMS (Degrees/Minutes/Seconds) text string, or NULL on failure

See also

[gaiaLongitudeFromDMS](#), [gaiaLatitudeFromDMS](#)

Note

this function will return a dynamically allocated buffer created by `malloc()`. You are required to explicitly `free()` any string returned by this function.

6.2.2.5 [gaiaConvertToUTF8\(\)](#)

```
GAIAAUX_DECLARE char* gaiaConvertToUTF8 (
    void * cvtCS,
    const char * buf,
    int len,
    int * err )
```

Converts a text string to UTF8.

Parameters

<i>cvtCS</i>	the handle identifying the UTF8 convert object (returned by a previous call to gaiaCreateUTF8Converter).
<i>buf</i>	the input text string
<i>len</i>	length (in bytes) of input string
<i>err</i>	on completion will contain 0 on success, any other value on failure

Returns

the null-terminated UTF8 encoded string: NULL on failure

See also

[gaiaCreateUTF8Converter](#), [gaiaFreeUTF8Converter](#)

Note

this function can safely handle strings of arbitrary length, and will return the converted string into a dynamically allocated buffer created by malloc(). You are required to explicitly free() any string returned by this function.

6.2.2.6 gaiaCreateMD5Checksum()

```
GAIAAUX_DECLARE void* gaiaCreateMD5Checksum (
    void )
```

Creates a persistent MD5 checksum object.

Returns

the handle of an MD5 checksum object, or NULL on failure

See also

[gaiaFreeMD5Checksum](#), [gaiaUpdateMD5Checksum](#), [gaiaFinalizeMD5Checksum](#)

Note

you must properly destroy the MD5 object when it isn't any longer used.

6.2.2.7 gaiaCreateUTF8Converter()

```
GAIAAUX_DECLARE void* gaiaCreateUTF8Converter (
    const char * fromCS )
```

Creates a persistent UTF8 converter object.

Parameters

<i>fromCS</i>	the GNU ICONV name identifying the input charset
---------------	--

Returns

the handle of the converter object, or NULL on failure

See also

[gaiaFreeUTF8Converter](#)

Note

you must properly destroy the converter object when it isn't any longer used.

6.2.2.8 `gaiaDecodeURL()`

```
GAIAAUX_DECLARE char* gaiaDecodeURL (
    const char * encoded,
    const char * out_charset )
```

Return a clean URL from its percent-encoded representation.

Parameters

<i>encoded</i>	the percent-encoded URL to be decoded
<i>in_charset</i>	the charset encoding adopted by the URL to be decoded

Returns

the corresponding clean URL text string, or NULL on failure

See also

[gaiaEncodeURL](#)

Note

this function will return a dynamically allocated buffer created by `malloc()`. You are required to explicitly `free()` any string returned by this function.

6.2.2.9 `gaiaDequotedSql()`

```
GAIAAUX_DECLARE char* gaiaDequotedSql (
    const char * value )
```

Properly formats an SQL generic string (dequoting)

Parameters

<i>value</i>	the string to be dequoted
--------------	---------------------------

Returns

the formatted string: NULL on failure

See also

[gaiaSingleQuotedSql](#), [gaiaDoubleQuotedSql](#), [gaiaQuotedSql](#)

Note

this function can safely handle strings of arbitrary length, and will return the formatted string into a dynamically allocated buffer created by malloc(). You are required to explicitly free() any string returned by this function.

6.2.2.10 gaiaDirNameFromPath()

```
GAIAUX_DECLARE char* gaiaDirNameFromPath (  
    const char * path )
```

Return the DirName component (if any) from a Path.

Parameters

<i>path</i>	full or relative pathname
-------------	---------------------------

Returns

the corresponding DirName text string, or NULL on failure

See also

[gaiaFullFileNameFromPath](#), [gaiaFileNameFromPath](#), [gaiaFileExtFromPath](#)

Note

this function will return a dynamically allocated buffer created by malloc(). You are required to explicitly free() any string returned by this function.

6.2.2.11 gaiaDoubleQuotedSql()

```
GAIAUX_DECLARE char* gaiaDoubleQuotedSql (  
    const char * value )
```

Properly formats an SQL name.

Parameters

<i>value</i>	the SQL name to be formatted
--------------	------------------------------

Returns

the formatted string: NULL on failure

See also

[gaiaQuotedSql](#), [gaiaDequotedSql](#)

Note

this function simply is a convenience method corresponding to: `gaiaQuotedSQL(value, GAIA_SQL_↵DOUBLE_QUOTE);`

Remarks

passing a string like "Sant\"Andrea" will return "Sant""Andrea"

6.2.2.12 gaiaEncodeURL()

```
GAIAAUX_DECLARE char* gaiaEncodeURL (  
    const char * url,  
    const char * in_charset )
```

Return a percent-encoded URL.

Parameters

<i>url</i>	the URL to be percent-encoded
<i>out_charset</i>	the charset encoding adopted by the encoded URL

Returns

the corresponding percent-encoded URL text string, or NULL on failure

See also

[gaiaDecodeURL](#)

Note

this function will return a dynamically allocated buffer created by `malloc()`. You are required to explicitly `free()` any string returned by this function.

6.2.2.13 gaiaFileExtFromPath()

```
GAIAAUX_DECLARE char* gaiaFileExtFromPath (  
    const char * path )
```

Return the FileExtension from a Path.

Parameters

<i>path</i>	full or relative pathname
-------------	---------------------------

Returns

the corresponding FileExtension (if any), or NULL on failure

See also

[gaiaDirNameFromPath](#), [gaiaFullFileNameFromPath](#), [gaiaFileNameFromPath](#)

Note

this function will return a dynamically allocated buffer created by malloc(). You are required to explicitly free() any string returned by this function.

6.2.2.14 gaiaFileNameFromPath()

```
GAIAAUX_DECLARE char* gaiaFileNameFromPath (  
    const char * path )
```

Return the FileName from a Path.

Parameters

<i>path</i>	full or relative pathname
-------------	---------------------------

Returns

the corresponding FileName (excluding an eventual extension), or NULL on failure

See also

[gaiaDirNameFromPath](#), [gaiaFullFileNameFromPath](#), [gaiaFileExtFromPath](#)

Note

this function will return a dynamically allocated buffer created by malloc(). You are required to explicitly free() any string returned by this function.

6.2.2.15 gaiaFinalizeMD5Checksum()

```
GAIAAUX_DECLARE char* gaiaFinalizeMD5Checksum (  
    void * md5 )
```

Return an MD5 checksum value.

Parameters

<i>md5</i>	the handle of the MD5 checksum object (returned by a previous call to <code>gaiaCreateMD5Checksum</code>).
------------	---

Returns

an hexadecimal text string representing the MD checksum: NULL on failure

See also

[gaiaCreateMD5Checksum](#), [gaiaUpdateMD5Checksum](#), [gaiaFreeMD5Checksum](#)

Note

this function will return the MD5 checksum into a dynamically allocated buffer created by `malloc()`. You are required to explicitly `free()` any string returned by this function.

`gaiaFinalizeMD5Checksum` will implicitly reset the MD5 object to its initial state.

6.2.2.16 `gaiaFreeMD5Checksum()`

```
GAIAAUX_DECLARE void gaiaFreeMD5Checksum (
    void * md5 )
```

Destroys an MD5 checksum object.

Parameters

<i>md5</i>	the handle of the MD5 checksum object (returned by a previous call to <code>gaiaCreateMD5Checksum</code>).
------------	---

See also

[gaiaCreateMD5Checksum](#)

6.2.2.17 `gaiaFreeUTF8Converter()`

```
GAIAAUX_DECLARE void gaiaFreeUTF8Converter (
    void * cvtCS )
```

Destroys an UTF8 converter object.

Parameters

<i>cvtCS</i>	the handle identifying the UTF8 convert object (returned by a previous call to <code>gaiaCreateUTF8Converter</code>).
--------------	--

See also

[gaiaCreateUTF8Converter](#)

6.2.2.18 `gaiaFullFileNameFromPath()`

```
GAIAAUX_DECLARE char* gaiaFullFileNameFromPath (  
    const char * path )
```

Return the FullFileName from a Path.

Parameters

<i>path</i>	full or relative pathname
-------------	---------------------------

Returns

the corresponding FullFileName (including an eventual extension), or NULL on failure

See also

[gaiaDirNameFromPath](#), [gaiaFileNameFromPath](#), [gaiaFileExtFromPath](#)

Note

this function will return a dynamically allocated buffer created by malloc(). You are required to explicitly free() any string returned by this function.

6.2.2.19 `gaiaGetLocaleCharset()`

```
GAIAAUX_DECLARE const char* gaiaGetLocaleCharset (  
    void )
```

Retrieves the Locale Charset.

Returns

the GNU ICONV name identifying the locale charset

6.2.2.20 `gaiaIllegalSqlName()`

```
GAIAAUX_DECLARE int gaiaIllegalSqlName (  
    const char * name )
```

Checks if a name is an illegal SQL name.

Parameters

<i>name</i>	the name to be checked
-------------	------------------------

Returns

0 if no: any other value if yes

See also

[gaiaIsReservedSqliteName](#), [gaiaIsReservedSqlName](#)

6.2.2.21 gaiaInsertIntoSqlLog()

```
GAIAAUX_DECLARE void gaiaInsertIntoSqlLog (
    sqlite3 * sqlite,
    const char * user_agent,
    const char * utf8Sql,
    sqlite3_int64 * sqllog_pk )
```

SQL log: statement start.

Parameters

<i>sqlite</i>	handle of the current DB connection
<i>user_agent</i>	name of the invoking application, e.g. "spatialite_gui" or "spatialite CLI"
<i>utf8Sql</i>	the SQL statement bein executed
<i>sqllog_pk</i>	after completion this variable will contain the value of the Primary Key identifying the corresponding Log event

See also

[gaiaUpdateSqlLog](#)

Note

this function inserts an **event** into the SQL Log, and is expected to be invoked immediately **before** executing the SQL statement itself.

6.2.2.22 gaiaIsReservedSqliteName()

```
GAIAAUX_DECLARE int gaiaIsReservedSqliteName (
    const char * name )
```

Checks if a name is a reserved SQLite name.

Parameters

<i>name</i>	the name to be checked
-------------	------------------------

Returns

0 if no: any other value if yes

See also

[gaiaIsReservedSqlName](#), [gaiaIllegalSqlName](#)

6.2.2.23 gaiaIsReservedSqlName()

```
GAIAAUX_DECLARE int gaiaIsReservedSqlName (  
    const char * name )
```

Checks if a name is a reserved SQL name.

Parameters

<i>name</i>	the name to be checked
-------------	------------------------

Returns

0 if no: any other value if yes

See also

[gaiaIsReservedSqliteName](#), [gaiaIllegalSqlName](#)

6.2.2.24 gaiaParseDMS()

```
GAIAAUX_DECLARE int gaiaParseDMS (  
    const char * dms,  
    double * longitude,  
    double * latitude )
```

Return longitude and latitude angles from a DMS string.

Parameters

<i>dms</i>	a text string representing a valid DMS (Degrees/Minutes/Seconds) expression.
<i>longitude</i>	on completion this variable will contain the longitude angle expressed in Decimal Degrees.
<i>latitude</i>	on completion this variable will contain the latitude angle expressed in Decimal Degrees.

Returns

ZERO (FALSE) on failure, any other different value (TRUE) on success.

See also

[gaiaConvertToDMS](#)

Note

this function will return a dynamically allocated buffer created by malloc(). You are required to explicitly free() any string returned by this function.

6.2.2.25 gaiaQuotedSql()

```
GAIAAUX_DECLARE char* gaiaQuotedSql (
    const char * value,
    int quote )
```

Properly formats an SQL generic string.

Parameters

<i>value</i>	the string to be formatted
<i>quote</i>	GAIA_SQL_SINGLE_QUOTE or GAIA_SQL_DOUBLE_QUOTE

Returns

the formatted string: NULL on failure

See also

[gaiaSingleQuotedSql](#), [gaiaDoubleQuotedSql](#), [gaiaDequotedSql](#)

Note

this function can safely handle strings of arbitrary length, and will return the formatted string into a dynamically allocated buffer created by malloc(). You are required to explicitly free() any string returned by this function.

6.2.2.26 gaiaRemoveExtraSpaces()

```
GAIAAUX_DECLARE char* gaiaRemoveExtraSpaces (
    const char * string )
```

Return a text string containing no repeated whitespaces.

Parameters

<i>string</i>	the input string to be cleaned
---------------	--------------------------------

Returns

the corresponding string containing no repeated whitespaces, or NULL on failure

Note

this function will return a dynamically allocated buffer created by malloc(). You are required to explicitly free() any string returned by this function.

6.2.2.27 gaiaSingleQuotedSql()

```
GAIAAUX_DECLARE char* gaiaSingleQuotedSql (  
    const char * value )
```

Properly formats an SQL text constant.

Parameters

<i>value</i>	the text string to be formatted
--------------	---------------------------------

Returns

the formatted string: NULL on failure

See also

[gaiaQuotedSql](#), [gaiaDequotedSql](#)

Note

this function simply is a convenience method corresponding to: `gaiaQuotedSQL(value, GAIA_SQL_SINGLE↵_QUOTE);`

Remarks

passing a string like "Sant'Andrea" will return 'Sant"Andrea'

6.2.2.28 gaiaUpdateMD5Checksum()

```
GAIAAUX_DECLARE void gaiaUpdateMD5Checksum (  
    void * md5,  
    const unsigned char * blob,  
    int blob_len )
```

Updates an MD5 checksum object.

Parameters

<i>md5</i>	the handle of the MD5 checksum object (returned by a previous call to <code>gaiaCreateMD5Checksum</code>).
<i>blob</i>	an arbitrary sequence of binary data
<i>blob_size</i>	the length (in bytes) of the binary data

See also

[gaiaCreateMD5Checksum](#), [gaiaFreeMD5Checksum](#), [gaiaFinalizeMD5Checksum](#)

Note

you can repeatedly invoke `gaiaUpdateMD5Checksum` more than a single time and always using the same MD5 object. In this case the final MD5 checksum returned by `gaiaGetMD5Checksum` will be the total checksum for any data processed by the MD5 object since its initialization.

6.2.2.29 `gaiaUpdateSqlLog()`

```
GAIAAUX_DECLARE void gaiaUpdateSqlLog (
    sqlite3 * sqlite,
    sqlite3_int64 sqllog_pk,
    int success,
    const char * errMsg )
```

SQL log: statement start.

Parameters

<i>sqlite</i>	handle of the current DB connection
<i>sqllog_pk</i>	the Primary Key identifying the corresponding Log event. expected to be exactly the same returned by the most recent call to gaiaInsertIntoSqlLog()
<i>success</i>	expected to be TRUE if the SQL statement was successfully executed.
<i>errMsg</i>	expected to be the error message returned by SQLite on failure, NULL on success.

See also

[gaiaInsertIntoSqlLog](#)

Note

this function completes an **event** inserted into the SQL Log, and is expected to be invoked immediately **after** executing the SQL statement itself.

6.3 `src/headers/spatialite/gaiaexif.h` File Reference

EXIF/image: supporting functions and constants.

Data Structures

- struct [gaiaExifTagStruct](#)
Container for an EXIF tag.
- struct [gaiaExifTagListStruct](#)
Container for a list of EXIF tags.

Macros

- #define [GAIA_HEX_BLOB](#) 0
generic hexadecimal BLOB
- #define [GAIA_GIF_BLOB](#) 1
this BLOB does actually contain a GIF image
- #define [GAIA_PNG_BLOB](#) 2
this BLOB does actually contain a PNG image
- #define [GAIA_JPEG_BLOB](#) 3
this BLOB does actually contain a generic JPEG image
- #define [GAIA_EXIF_BLOB](#) 4
this BLOB does actually contain a JPEG-EXIF image
- #define [GAIA_EXIF_GPS_BLOB](#) 5
this BLOB does actually contain a JPEG-EXIF image including GPS data
- #define [GAIA_ZIP_BLOB](#) 6
this BLOB does actually contain a ZIP compressed file
- #define [GAIA_PDF_BLOB](#) 7
this BLOB does actually contain a PDF document
- #define [GAIA_GEOMETRY_BLOB](#) 8
this BLOB does actually contain a SpatiaLite Geometry (not compressed)
- #define [GAIA_COMPRESSED_GEOMETRY_BLOB](#) 9
this BLOB does actually contain a SpatiaLite Geometry (compressed)
- #define [GAIA_TIFF_BLOB](#) 10
this BLOB does actually contain a TIFF image
- #define [GAIA_WEBP_BLOB](#) 11
this BLOB does actually contain a WebP image
- #define [GAIA_JP2_BLOB](#) 12
this BLOB does actually contain a JP2 (Jpeg2000) image
- #define [GAIA_XML_BLOB](#) 13
this BLOB does actually contain a SpatiaLite XmlBLOB
- #define [GAIA_GPB_BLOB](#) 14
this BLOB does actually contain a GPKG Geometry
- #define [GAIA_TINYPOINT_BLOB](#) 15
this BLOB does actually contain a SpatiaLite TinyPoint
- #define [GAIA_EXIF_NONE](#) 0
unrecognized EXIF value
- #define [GAIA_EXIF_BYTE](#) 1
EXIF value of the BYTE type.
- #define [GAIA_EXIF_SHORT](#) 2
EXIF value of the SHORT type.
- #define [GAIA_EXIF_STRING](#) 3
EXIF value of the STRING type.
- #define [GAIA_EXIF_LONG](#) 4

- *EXIF value of the LONG type.*
- #define `GAIA_EXIF_RATIONAL` 5
EXIF value of the RATIONAL type.
- #define `GAIA_EXIF_SLONG` 9
EXIF value of the SLONG type.
- #define `GAIA_EXIF_SRATIONAL` 10
EXIF value of the SRATIONAL type.

Typedefs

- typedef struct `gaiaExifTagStruct` `gaiaExifTag`
Container for an EXIF tag.
- typedef `gaiaExifTag` * `gaiaExifTagPtr`
Typedef for EXIF tag structure.
- typedef struct `gaiaExifTagListStruct` `gaiaExifTagList`
Container for a list of EXIF tags.
- typedef `gaiaExifTagList` * `gaiaExifTagListPtr`
Typedef for EXIF tag structure.

Functions

- GAIAEXIF_DECLARE `gaiaExifTagListPtr` `gaiaGetExifTags` (const unsigned char *blob, int size)
Creates a list of EXIF tags by parsing a BLOB of the JPEG-EXIF type.
- GAIAEXIF_DECLARE void `gaiaExifTagsFree` (`gaiaExifTagListPtr` tag_list)
Destroy a list of EXIF tags.
- GAIAEXIF_DECLARE int `gaiaGetExifTagsCount` (`gaiaExifTagListPtr` tag_list)
Return the total number of EXIF tags into the list.
- GAIAEXIF_DECLARE `gaiaExifTagPtr` `gaiaGetExifTagByPos` (`gaiaExifTagListPtr` tag_list, const int pos)
Retrieves an EXIF tag by its relative position into the list.
- GAIAEXIF_DECLARE `gaiaExifTagPtr` `gaiaGetExifTagById` (const `gaiaExifTagListPtr` tag_list, const unsigned short tag_id)
Retrieves an EXIF tag by its Tag ID.
- GAIAEXIF_DECLARE `gaiaExifTagPtr` `gaiaGetExifGpsTagById` (const `gaiaExifTagListPtr` tag_list, const unsigned short tag_id)
Retrieves an EXIF-GPS tag by its Tag ID.
- GAIAEXIF_DECLARE `gaiaExifTagPtr` `gaiaGetExifTagByName` (const `gaiaExifTagListPtr` tag_list, const char *tag_name)
Retrieves an EXIF tag by its name.
- GAIAEXIF_DECLARE unsigned short `gaiaExifTagGetId` (const `gaiaExifTagPtr` tag)
Return the Tag ID from an EXIF tag.
- GAIAEXIF_DECLARE void `gaiaExifTagGetName` (const `gaiaExifTagPtr` tag, char *tag_name, int len)
Return the Tag Name from an EXIF tag.
- GAIAEXIF_DECLARE int `gaiaIsExifGpsTag` (const `gaiaExifTagPtr` tag)
Checks if an EXIF tag actually is an EXIF-GPS tag.
- GAIAEXIF_DECLARE unsigned short `gaiaExifTagGetValueType` (const `gaiaExifTagPtr` tag)
Return the value type for an EXIF tag.
- GAIAEXIF_DECLARE unsigned short `gaiaExifTagGetNumValues` (const `gaiaExifTagPtr` tag)
Return the total count of values from an EXIF tag.
- GAIAEXIF_DECLARE unsigned char `gaiaExifTagGetByteValue` (const `gaiaExifTagPtr` tag, const int ind, int *ok)

- Return a BYTE value from an EXIF tag.*
- GAIAEXIF_DECLARE void [gaiaExifTagGetStringValue](#) (const [gaiaExifTagPtr](#) tag, char *str, int len, int *ok)
- Return a STRING value from an EXIF tag.*
- GAIAEXIF_DECLARE unsigned short [gaiaExifTagGetShortValue](#) (const [gaiaExifTagPtr](#) tag, const int ind, int *ok)
- Return a SHORT value from an EXIF tag.*
- GAIAEXIF_DECLARE unsigned int [gaiaExifTagGetLongValue](#) (const [gaiaExifTagPtr](#) tag, const int ind, int *ok)
- Return a LONG value from an EXIF tag.*
- GAIAEXIF_DECLARE unsigned int [gaiaExifTagGetRational1Value](#) (const [gaiaExifTagPtr](#) tag, const int ind, int *ok)
- Return a RATIONAL [numerator] value from an EXIF tag.*
- GAIAEXIF_DECLARE unsigned int [gaiaExifTagGetRational2Value](#) (const [gaiaExifTagPtr](#) tag, const int ind, int *ok)
- Return a RATIONAL [denominator] value from an EXIF tag.*
- GAIAEXIF_DECLARE double [gaiaExifTagGetRationalValue](#) (const [gaiaExifTagPtr](#) tag, const int ind, int *ok)
- Return a RATIONAL value from an EXIF tag.*
- GAIAEXIF_DECLARE short [gaiaExifTagGetSignedShortValue](#) (const [gaiaExifTagPtr](#) tag, const int ind, int *ok)
- Return a Signed SHORT value from an EXIF tag.*
- GAIAEXIF_DECLARE int [gaiaExifTagGetSignedLongValue](#) (const [gaiaExifTagPtr](#) tag, const int ind, int *ok)
- Return a Signed LONG value from an EXIF tag.*
- GAIAEXIF_DECLARE int [gaiaExifTagGetSignedRational1Value](#) (const [gaiaExifTagPtr](#) tag, const int ind, int *ok)
- Return a SRATIONAL [numerator] value from an EXIF tag.*
- GAIAEXIF_DECLARE int [gaiaExifTagGetSignedRational2Value](#) (const [gaiaExifTagPtr](#) tag, const int ind, int *ok)
- Return a SRATIONAL [denominator] value from an EXIF tag.*
- GAIAEXIF_DECLARE double [gaiaExifTagGetSignedRationalValue](#) (const [gaiaExifTagPtr](#) tag, const int ind, int *ok)
- Return a Signed RATIONAL value from an EXIF tag.*
- GAIAEXIF_DECLARE float [gaiaExifTagGetFloatValue](#) (const [gaiaExifTagPtr](#) tag, const int ind, int *ok)
- Return a FLOAT value from an EXIF tag.*
- GAIAEXIF_DECLARE double [gaiaExifTagGetDoubleValue](#) (const [gaiaExifTagPtr](#) tag, const int ind, int *ok)
- Return a DOUBLE value from an EXIF tag.*
- GAIAEXIF_DECLARE void [gaiaExifTagGetHumanReadable](#) (const [gaiaExifTagPtr](#) tag, char *str, int len, int *ok)
- Return a human readable description from an EXIF tag.*
- GAIAEXIF_DECLARE int [gaiaGuessBlobType](#) (const unsigned char *blob, int size)
- Attempts to guess the actual content-type of some BLOB.*
- GAIAEXIF_DECLARE int [gaiaGetGpsCoords](#) (const unsigned char *blob, int size, double *longitude, double *latitude)
- Return longitude and latitude from an EXIF-GPS tag.*
- GAIAEXIF_DECLARE int [gaiaGetGpsLatLong](#) (const unsigned char *blob, int size, char *latlong, int ll_size)
- Return a text string representing DMS coordinates from an EXIF-GPS tag.*

6.3.1 Detailed Description

EXIF/image: supporting functions and constants.

6.3.2 Typedef Documentation

6.3.2.1 `gaiaExifTagListPtr`

```
typedef gaiaExifTagList\* gaiaExifTagListPtr
```

Typedef for EXIF tag structure.

See also

[gaiaExifTagListStruct](#)

6.3.2.2 `gaiaExifTagPtr`

```
typedef gaiaExifTag\* gaiaExifTagPtr
```

Typedef for EXIF tag structure.

See also

[gaiaExifTagStruct](#)

6.3.3 Function Documentation

6.3.3.1 `gaiaExifTagGetByteValue()`

```
GAIAEXIF_DECLARE unsigned char gaiaExifTagGetByteValue (  
    const gaiaExifTagPtr tag,  
    const int ind,  
    int * ok )
```

Return a BYTE value from an EXIF tag.

Parameters

<i>tag</i>	pointer to an EXIF tag.
<i>ind</i>	value index [first value has index 0].
<i>ok</i>	on completion will contain 0 on failure: any other value on success.

Returns

the BYTE value

See also

[gaiaGetExifTagById](#), [gaiaGetExifGpsTagById](#), [gaiaGetExifTagByName](#), [gaiaExifTagGetValueType](#), [gaiaExifTagGetNumValues](#)

6.3.3.2 gaiaExifTagGetDoubleValue()

```
GAIAEXIF_DECLARE double gaiaExifTagGetDoubleValue (
    const gaiaExifTagPtr tag,
    const int ind,
    int * ok )
```

Return a DOUBLE value from an EXIF tag.

Parameters

<i>tag</i>	pointer to an EXIF tag.
<i>ind</i>	value index [first value has index 0].
<i>ok</i>	on completion will contain 0 on failure: any other value on success.

Returns

the DOUBLE value

See also

[gaiaGetExifTagById](#), [gaiaGetExifGpsTagById](#), [gaiaGetExifTagByName](#), [gaiaExifTagGetValueType](#), [gaiaExifTagGetNumValues](#)

6.3.3.3 gaiaExifTagGetFloatValue()

```
GAIAEXIF_DECLARE float gaiaExifTagGetFloatValue (
    const gaiaExifTagPtr tag,
    const int ind,
    int * ok )
```

Return a FLOAT value from an EXIF tag.

Parameters

<i>tag</i>	pointer to an EXIF tag.
<i>ind</i>	value index [first value has index 0].
<i>ok</i>	on completion will contain 0 on failure: any other value on success.

Returns

the FLOAT value

See also

[gaiaGetExifTagById](#), [gaiaGetExifGpsTagById](#), [gaiaGetExifTagByName](#), [gaiaExifTagGetValueType](#), [gaiaExifTagGetNumValues](#)

6.3.3.4 gaiaExifTagGetHumanReadable()

```
GAIAEXIF_DECLARE void gaiaExifTagGetHumanReadable (
    const gaiaExifTagPtr tag,
    char * str,
    int len,
    int * ok )
```

Return a human readable description from an EXIF tag.

Parameters

<i>tag</i>	pointer to an EXIF tag.
<i>str</i>	receiving buffer: the STRING value will be copied here.
<i>len</i>	length of the receiving buffer
<i>ok</i>	on completion will contain 0 on failure: any other value on success.

See also

[gaiaGetExifTagById](#), [gaiaGetExifGpsTagById](#), [gaiaGetExifTagByName](#)

6.3.3.5 gaiaExifTagGetId()

```
GAIAEXIF_DECLARE unsigned short gaiaExifTagGetId (
    const gaiaExifTagPtr tag )
```

Return the Tag ID from an EXIF tag.

Parameters

<i>tag</i>	pointer to an EXIF tag
------------	------------------------

Returns

the Tag ID

See also

[gaiaGetExifTagById](#), [gaiaGetExifGpsTagById](#), [gaiaGetExifTagByName](#)

6.3.3.6 `gaiaExifTagGetLongValue()`

```
GAIAEXIF_DECLARE unsigned int gaiaExifTagGetLongValue (
    const gaiaExifTagPtr tag,
    const int ind,
    int * ok )
```

Return a LONG value from an EXIF tag.

Parameters

<i>tag</i>	pointer to an EXIF tag.
<i>ind</i>	value index [first value has index 0].
<i>ok</i>	on completion will contain 0 on failure: any other value on success.

Returns

the LONG value

See also

[gaiaGetExifTagById](#), [gaiaGetExifGpsTagById](#), [gaiaGetExifTagByName](#), [gaiaExifTagGetValueType](#), [gaiaExifTagGetNumValues](#)

6.3.3.7 `gaiaExifTagGetName()`

```
GAIAEXIF_DECLARE void gaiaExifTagGetName (
    const gaiaExifTagPtr tag,
    char * tag_name,
    int len )
```

Return the Tag Name from an EXIF tag.

Parameters

<i>tag</i>	pointer to an EXIF tag
<i>tag_name</i>	receiving buffer: the Tag Name will be copied here
<i>len</i>	length of the receiving buffer

See also

[gaiaGetExifTagById](#), [gaiaGetExifGpsTagById](#), [gaiaGetExifTagByName](#)

6.3.3.8 `gaiaExifTagGetNumValues()`

```
GAIAEXIF_DECLARE unsigned short gaiaExifTagGetNumValues (
    const gaiaExifTagPtr tag )
```

Return the total count of values from an EXIF tag.

Parameters

<i>tag</i>	pointer to an EXIF tag
------------	------------------------

Returns

the number of available values

See also

[gaiaGetExifTagById](#), [gaiaGetExifGpsTagById](#), [gaiaGetExifTagByName](#)

6.3.3.9 `gaiaExifTagGetRational1Value()`

```
GAIAEXIF_DECLARE unsigned int gaiaExifTagGetRational1Value (
    const gaiaExifTagPtr tag,
    const int ind,
    int * ok )
```

Return a RATIONAL [numerator] value from an EXIF tag.

Parameters

<i>tag</i>	pointer to an EXIF tag.
<i>ind</i>	value index [first value has index 0].
<i>ok</i>	on completion will contain 0 on failure: any other value on success.

Returns

the RATIONAL [numerator] value

See also

[gaiaGetExifTagById](#), [gaiaGetExifGpsTagById](#), [gaiaGetExifTagByName](#), [gaiaExifTagGetValueType](#), [gaiaExifTagGetNumValues](#)

6.3.3.10 `gaiaExifTagGetRational2Value()`

```
GAIAEXIF_DECLARE unsigned int gaiaExifTagGetRational2Value (
    const gaiaExifTagPtr tag,
    const int ind,
    int * ok )
```

Return a RATIONAL [denominator] value from an EXIF tag.

Parameters

<i>tag</i>	pointer to an EXIF tag.
<i>ind</i>	value index [first value has index 0].
<i>ok</i>	on completion will contain 0 on failure: any other value on success.

Returns

the RATIONAL [denominator] value

See also

[gaiaGetExifTagById](#), [gaiaGetExifGpsTagById](#), [gaiaGetExifTagByName](#), [gaiaExifTagGetValueType](#), [gaiaExifTagGetNumValues](#)

6.3.3.11 `gaiaExifTagGetRationalValue()`

```
GAIAEXIF_DECLARE double gaiaExifTagGetRationalValue (
    const gaiaExifTagPtr tag,
    const int ind,
    int * ok )
```

Return a RATIONAL value from an EXIF tag.

Parameters

<i>tag</i>	pointer to an EXIF tag.
<i>ind</i>	value index [first value has index 0].
<i>ok</i>	on completion will contain 0 on failure: any other value on success.

Returns

the RATIONAL value

See also

[gaiaGetExifTagById](#), [gaiaGetExifGpsTagById](#), [gaiaGetExifTagByName](#), [gaiaExifTagGetValueType](#), [gaiaExifTagGetNumValues](#)

6.3.3.12 `gaiaExifTagGetShortValue()`

```
GAIAEXIF_DECLARE unsigned short gaiaExifTagGetShortValue (
    const gaiaExifTagPtr tag,
    const int ind,
    int * ok )
```

Return a SHORT value from an EXIF tag.

Parameters

<i>tag</i>	pointer to an EXIF tag.
<i>ind</i>	value index [first value has index 0].
<i>ok</i>	on completion will contain 0 on failure: any other value on success.

Returns

the SHORT value

See also

[gaiaGetExifTagById](#), [gaiaGetExifGpsTagById](#), [gaiaGetExifTagByName](#), [gaiaExifTagGetValueType](#), [gaiaExifTagGetNumValues](#)

6.3.3.13 `gaiaExifTagGetSignedLongValue()`

```
GAIAEXIF_DECLARE int gaiaExifTagGetSignedLongValue (
    const gaiaExifTagPtr tag,
    const int ind,
    int * ok )
```

Return a Signed LONG value from an EXIF tag.

Parameters

<i>tag</i>	pointer to an EXIF tag.
<i>ind</i>	value index [first value has index 0].
<i>ok</i>	on completion will contain 0 on failure: any other value on success.

Returns

the Signed LONG value

See also

[gaiaGetExifTagById](#), [gaiaGetExifGpsTagById](#), [gaiaGetExifTagByName](#), [gaiaExifTagGetValueType](#), [gaiaExifTagGetNumValues](#)

6.3.3.14 `gaiaExifTagGetSignedRational1Value()`

```
GAIAEXIF_DECLARE int gaiaExifTagGetSignedRational1Value (
    const gaiaExifTagPtr tag,
    const int ind,
    int * ok )
```

Return a SRATIONAL [numerator] value from an EXIF tag.

Parameters

<i>tag</i>	pointer to an EXIF tag.
<i>ind</i>	value index [first value has index 0].
<i>ok</i>	on completion will contain 0 on failure: any other value on success.

Returns

the SRATIONAL [numerator] value

See also

[gaiaGetExifTagById](#), [gaiaGetExifGpsTagById](#), [gaiaGetExifTagByName](#), [gaiaExifTagGetValueType](#), [gaiaExifTagGetNumValues](#)

6.3.3.15 `gaiaExifTagGetSignedRational2Value()`

```
GAIAEXIF_DECLARE int gaiaExifTagGetSignedRational2Value (
    const gaiaExifTagPtr tag,
    const int ind,
    int * ok )
```

Return a SRATIONAL [denominator] value from an EXIF tag.

Parameters

<i>tag</i>	pointer to an EXIF tag.
<i>ind</i>	value index [first value has index 0].
<i>ok</i>	on completion will contain 0 on failure: any other value on success.

Returns

the SRATIONAL [denominator] value

See also

[gaiaGetExifTagById](#), [gaiaGetExifGpsTagById](#), [gaiaGetExifTagByName](#), [gaiaExifTagGetValueType](#), [gaiaExifTagGetNumValues](#)

6.3.3.16 `gaiaExifTagGetSignedRationalValue()`

```
GAIAEXIF_DECLARE double gaiaExifTagGetSignedRationalValue (
    const gaiaExifTagPtr tag,
    const int ind,
    int * ok )
```

Return a Signed RATIONAL value from an EXIF tag.

Parameters

<i>tag</i>	pointer to an EXIF tag.
<i>ind</i>	value index [first value has index 0].
<i>ok</i>	on completion will contain 0 on failure: any other value on success.

Returns

the Signed RATIONAL value

See also

[gaiaGetExifTagById](#), [gaiaGetExifGpsTagById](#), [gaiaGetExifTagByName](#), [gaiaExifTagGetValueType](#), [gaiaExifTagGetNumValues](#)

6.3.3.17 `gaiaExifTagGetSignedShortValue()`

```
GAIAEXIF_DECLARE short gaiaExifTagGetSignedShortValue (
    const gaiaExifTagPtr tag,
    const int ind,
    int * ok )
```

Return a Signed SHORT value from an EXIF tag.

Parameters

<i>tag</i>	pointer to an EXIF tag.
<i>ind</i>	value index [first value has index 0].
<i>ok</i>	on completion will contain 0 on failure: any other value on success.

Returns

the Signed SHORT value

See also

[gaiaGetExifTagById](#), [gaiaGetExifGpsTagById](#), [gaiaGetExifTagByName](#), [gaiaExifTagGetValueType](#), [gaiaExifTagGetNumValues](#)

6.3.3.18 `gaiaExifTagGetStringValue()`

```
GAIAEXIF_DECLARE void gaiaExifTagGetStringValue (
    const gaiaExifTagPtr tag,
    char * str,
    int len,
    int * ok )
```

Return a STRING value from an EXIF tag.

Parameters

<i>tag</i>	pointer to an EXIF tag.
<i>str</i>	receiving buffer: the STRING value will be copied here.
<i>len</i>	length of the receiving buffer
<i>ok</i>	on completion will contain 0 on failure: any other value on success.

See also

[gaiaGetExifTagById](#), [gaiaGetExifGpsTagById](#), [gaiaGetExifTagByName](#), [gaiaExifTagGetValueType](#), [gaiaExifTagGetNumValues](#)

6.3.3.19 `gaiaExifTagGetValueType()`

```
GAIAEXIF_DECLARE unsigned short gaiaExifTagGetValueType (
    const gaiaExifTagPtr tag )
```

Return the value type for an EXIF tag.

Parameters

<i>tag</i>	pointer to an EXIF tag
------------	------------------------

Returns

the value type: one of GAIA_EXIF_NONE, GAIA_EXIF_BYTE, GAIA_EXIF_SHORT, GAIA_EXIF_STRING, GAIA_EXIF_LONG, GAIA_EXIF_RATIONAL, GAIA_EXIF_SLONG, GAIA_EXIF_SRATIONAL

See also

[gaiaGetExifTagById](#), [gaiaGetExifGpsTagById](#), [gaiaGetExifTagByName](#)

6.3.3.20 `gaiaExifTagsFree()`

```
GAIAEXIF_DECLARE void gaiaExifTagsFree (
    gaiaExifTagListPtr tag_list )
```

Destroy a list of EXIF tags.

Parameters

<i>tag_list</i>	the list to be destroyed
-----------------	--------------------------

See also[gaiaGetExifTags](#)**Note**

the pointer passed to this function must be one returned by a previous call to [gaiaGetExifTags](#)

6.3.3.21 [gaiaGetExifGpsTagById\(\)](#)

```
GAIAEXIF_DECLARE gaiaExifTagPtr gaiaGetExifGpsTagById (  
    const gaiaExifTagListPtr tag_list,  
    const unsigned short tag_id )
```

Retrieves an EXIF-GPS tag by its Tag ID.

Parameters

<i>tag_list</i>	pointer to an EXIF tag list.
<i>tag_id</i>	the GPS Tag ID to be found

Returns

a pointer to the corresponding EXIF tag: NULL if not found

See also[gaiaGetExifTags](#), [gaiaExifTagsFree](#)**6.3.3.22 [gaiaGetExifTagById\(\)](#)**

```
GAIAEXIF_DECLARE gaiaExifTagPtr gaiaGetExifTagById (  
    const gaiaExifTagListPtr tag_list,  
    const unsigned short tag_id )
```

Retrieves an EXIF tag by its Tag ID.

Parameters

<i>tag_list</i>	pointer to an EXIF tag list.
<i>tag_id</i>	the Tag ID to be found

Returns

a pointer to the corresponding EXIF tag: NULL if not found

See also

[gaiaGetExifTags](#), [gaiaExifTagsFree](#)

6.3.3.23 gaiaGetExifTagByName()

```
GAIAEXIF_DECLARE gaiaExifTagPtr gaiaGetExifTagByName (
    const gaiaExifTagListPtr tag_list,
    const char * tag_name )
```

Retrieves an EXIF tag by its name.

Parameters

<i>tag_list</i>	pointer to an EXIF tag list.
<i>tag_name</i>	the Tag Name to be found

Returns

a pointer to the corresponding EXIF tag: NULL if not found

See also

[gaiaGetExifTags](#), [gaiaExifTagsFree](#)

6.3.3.24 gaiaGetExifTagByPos()

```
GAIAEXIF_DECLARE gaiaExifTagPtr gaiaGetExifTagByPos (
    gaiaExifTagListPtr tag_list,
    const int pos )
```

Retrieves an EXIF tag by its relative position into the list.

Parameters

<i>tag_list</i>	pointer to an EXIF tag list.
<i>pos</i>	relative item position [first item is 0]

Returns

a pointer to the corresponding EXIF tag: NULL if not found

See also

[gaiaGetExifTags](#), [gaiaExifTagsFree](#), [gaiaExifTagsCount](#)

6.3.3.25 gaiaGetExifTags()

```
GAIAEXIF_DECLARE gaiaExifTagListPtr gaiaGetExifTags (
    const unsigned char * blob,
    int size )
```

Creates a list of EXIF tags by parsing a BLOB of the JPEG-EXIF type.

Parameters

<i>blob</i>	the BLOB to be parsed
<i>size</i>	the BLOB size (in bytes)

Returns

a list of EXIF tags: or NULL if any error is encountered

See also

[gaiaExifTagsFree](#)

Note

you must explicitly destroy the list when it's any longer used.

6.3.3.26 gaiaGetExifTagsCount()

```
GAIAEXIF_DECLARE int gaiaGetExifTagsCount (
    gaiaExifTagListPtr tag_list )
```

Return the total number of EXIF tags into the list.

Parameters

<i>tag_list</i>	pointer to an EXIF tag list.
-----------------	------------------------------

Returns

the EXIF tag count.

See also

[gaiaGetExifTags](#), [gaiaExifTagsFree](#)

6.3.3.27 gaiaGetGpsCoords()

```
GAIAEXIF_DECLARE int gaiaGetGpsCoords (
    const unsigned char * blob,
    int size,
    double * longitude,
    double * latitude )
```

Return longitude and latitude from an EXIF-GPS tag.

Parameters

<i>blob</i>	the BLOB to be parsed
<i>size</i>	length of the BLOB (in bytes)
<i>longitude</i>	on success will contain the longitude coordinate
<i>latitude</i>	on success will contain the latitude coordinate

Returns

0 on failure: any other value on success

See also

[gaiaGetExifTagById](#), [gaiaGetExifGpsTagById](#), [gaiaGetExifTagByName](#), [gaiaIsExifGpsTag](#)

6.3.3.28 gaiaGetGpsLatLong()

```
GAIAEXIF_DECLARE int gaiaGetGpsLatLong (
    const unsigned char * blob,
    int size,
    char * latlong,
    int ll_size )
```

Return a text string representing DMS coordinates from an EXIF-GPS tag.

Parameters

<i>blob</i>	the BLOB to be parsed
<i>size</i>	length of the BLOB (in bytes)
<i>latlong</i>	receiving buffer: the text string will be copied here.
<i>ll_size</i>	length of the receiving buffer

Returns

0 on failure: any other value on success

See also

[gaiaGetExifTagById](#), [gaiaGetExifGpsTagById](#), [gaiaGetExifTagByName](#), [gaiaIsExifGpsTag](#)

6.3.3.29 gaiaGuessBlobType()

```
GAIAEXIF_DECLARE int gaiaGuessBlobType (
    const unsigned char * blob,
    int size )
```

Attempts to guess the actual content-type of some BLOB.

Parameters

<i>blob</i>	the BLOB to be parsed
<i>size</i>	length of the BLOB (in bytes)

Returns

the BLOB type: one of GAIA_HEX_BLOB, GAIA_GIF_BLOB, GAIA_PNG_BLOB, GAIA_JPEG_BLOB, GAIA_EXIF_BLOB, GAIA_EXIF_GPS_BLOB, GAIA_ZIP_BLOB, GAIA_PDF_BLOB, GAIA_GEOMETRY_BLOB, GAIA_TIFF_BLOB, GAIA_WEBP_BLOB, GAIA_JP2_BLOB, GAIA_XML_BLOB, GAIA_GPB_BLOB

6.3.3.30 gaiaIsExifGpsTag()

```
GAIAEXIF_DECLARE int gaiaIsExifGpsTag (
    const gaiaExifTagPtr tag )
```

Checks if an EXIF tag actually is an EXIF-GPS tag.

Parameters

<i>tag</i>	pointer to an EXIF tag
------------	------------------------

Returns

0 if false: any other value if true

See also

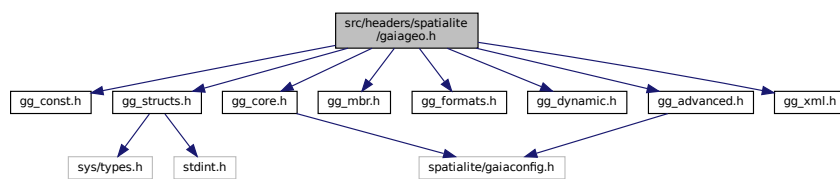
[gaiaGetExifTagById](#), [gaiaGetExifGpsTagById](#), [gaiaGetExifTagByName](#)

6.4 src/headers/spatialite/gaiageo.h File Reference

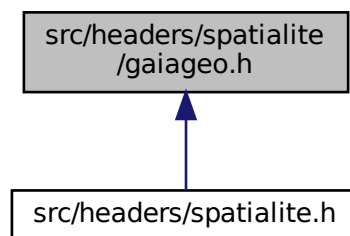
Geometry handling functions and constants.

```
#include "gg_const.h"
#include "gg_structs.h"
#include "gg_core.h"
#include "gg_mbr.h"
#include "gg_formats.h"
#include "gg_dynamic.h"
#include "gg_advanced.h"
#include "gg_xml.h"
```

Include dependency graph for gaiageo.h:



This graph shows which files directly or indirectly include this file:



6.4.1 Detailed Description

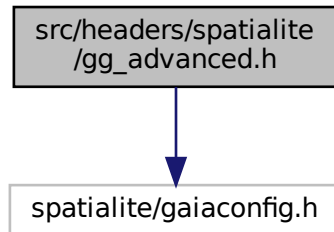
Geometry handling functions and constants.

6.5 src/headers/spatialite/gg_advanced.h File Reference

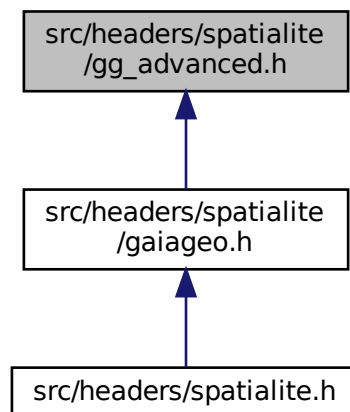
Geometry handling functions: advanced.

```
#include <spatialite/gaiageo.h>
```

Include dependency graph for `gg_advanced.h`:



This graph shows which files directly or indirectly include this file:



Macros

- `#define GAIA2GEOS_ALL 0`
Gaia-to-GEOS: all geometries.
- `#define GAIA2GEOS_ONLY_POINTS 1`
Gaia-to-GEOS: only geometries of the Point type.
- `#define GAIA2GEOS_ONLY_LINESTRINGS 2`
Gaia-to-GEOS: only geometries of the Linestring type.
- `#define GAIA2GEOS_ONLY_POLYGONS 3`
Gaia-to-GEOS: only geometries of the Polygon type.

Functions

- GAIAGEO_DECLARE void [gaiaResetProjErrorMsg_r](#) (const void *p_cache)
Resets the PROJ error messages to an empty state.
- GAIAGEO_DECLARE void [gaiaSetProjErrorMsg_r](#) (const void *p_cache, const char *msg)
Set the current PROJ error message.
- GAIAGEO_DECLARE const char * [gaiaGetProjErrorMsg_r](#) (const void *p_cache)
Return the latest PROJ error message (if any)
- GAIAGEO_DECLARE const char * [gaiaSetProjDatabasePath](#) (const void *p_cache, const char *path)
Sets the PATH leading to the private PROJ.6 database.
- GAIAGEO_DECLARE const char * [gaiaGetProjDatabasePath](#) (const void *p_cache)
Return the currently set PATH leading to the private PROJ.6 database.
- GAIAGEO_DECLARE char * [gaiaGetProjString](#) (const void *p_cache, const char *auth_name, int auth_srid)
Return the proj-string corresponding to a given CRS defined within the private PROJ.6 database.
- GAIAGEO_DECLARE char * [gaiaGetProjWKT](#) (const void *p_cache, const char *auth_name, int auth_srid, int style, int indented, int indentation)
Return the WKT expression corresponding to a given CRS defined within the private PROJ.6 database.
- GAIAGEO_DECLARE int [gaiaGuessSridFromWKT](#) (sqlite3 *db_handle, const void *p_cache, const char *wkt, int *srid)
Return the SRID corresponding to a given WKT expression (if any)
- GAIAGEO_DECLARE double [gaiaRadsToDegs](#) (double rads)
Converts and angle from Radians into Degrees.
- GAIAGEO_DECLARE double [gaiaDegsToRads](#) (double degs)
Converts and angle from Degrees into Radians.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaTransform](#) ([gaiaGeomCollPtr](#) org, const char *proj_from, const char *proj_to)
Transforms a Geometry object into a different Reference System [aka Reprojection].
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaTransform_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) org, const char *proj_from, const char *proj_to)
Transforms a Geometry object into a different Reference System [aka Reprojection].
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaTransformEx](#) ([gaiaGeomCollPtr](#) org, const char *proj_string_1, const char *proj_string_2, [gaiaProjAreaPtr](#) proj_bbox)
Transforms a Geometry object into a different Reference System [aka Reprojection].
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaTransformEx_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) org, const char *proj_string_1, const char *proj_string_2, [gaiaProjAreaPtr](#) proj_bbox)
Transforms a Geometry object into a different Reference System [aka Reprojection].
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaTransformXY](#) ([gaiaGeomCollPtr](#) org, const char *proj_from, const char *proj_to)
Transforms a Geometry object into a different Reference System [aka Reprojection] This is a special "flavor" of [gaiaTransform\(\)](#) just considering X and Y coordinates; Z and M values will be left untouched.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaTransformXY_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) org, const char *proj_from, const char *proj_to)
Transforms a Geometry object into a different Reference System [aka Reprojection] This is a special "flavor" of [gaiaTransform_r\(\)](#) just considering X and Y coordinates; Z and M values will be left untouched.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaTransformXYZ](#) ([gaiaGeomCollPtr](#) org, const char *proj_from, const char *proj_to)
Transforms a Geometry object into a different Reference System [aka Reprojection] This is a special "flavor" of [gaiaTransform\(\)](#) just considering X, Y and Z coordinates; M values will be left untouched.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaTransformXYZ_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) org, const char *proj_from, const char *proj_to)
Transforms a Geometry object into a different Reference System [aka Reprojection] This is a special "flavor" of [gaiaTransform_r\(\)](#) just considering X, Y and Z coordinates; M values will be left untouched.
- GAIAGEO_DECLARE void [gaiaResetGeosMsg](#) (void)

- Resets the GEOS error and warning messages to an empty state.*
- GAIAGEO_DECLARE void [gaiaResetGeosMsg_r](#) (const void *p_cache)
- Resets the GEOS error and warning messages to an empty state.*
- GAIAGEO_DECLARE const char * [gaiaGetGeosErrorMsg](#) (void)
- Return the latest GEOS error message (if any)*
- GAIAGEO_DECLARE const char * [gaiaGetGeosErrorMsg_r](#) (const void *p_cache)
- Return the latest GEOS error message (if any)*
- GAIAGEO_DECLARE const char * [gaiaGetGeosWarningMsg](#) (void)
- Return the latest GEOS warning message (if any)*
- GAIAGEO_DECLARE const char * [gaiaGetGeosWarningMsg_r](#) (const void *p_cache)
- Return the latest GEOS warning message (if any)*
- GAIAGEO_DECLARE const char * [gaiaGetGeosAuxErrorMsg](#) (void)
- Return the latest GEOS (auxiliary) error message (if any)*
- GAIAGEO_DECLARE const char * [gaiaGetGeosAuxErrorMsg_r](#) (const void *p_cache)
- Return the latest GEOS (auxiliary) error message (if any)*
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaCriticalPointFromGEOSmsg](#) (void)
- Attempts to (possible) return a Point Geometry extracted from the latest GEOS error / warning message.*
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaCriticalPointFromGEOSmsg_r](#) (const void *p_cache)
- Attempts to (possible) return a Point Geometry extracted from the latest GEOS error / warning message.*
- GAIAGEO_DECLARE void [gaiaSetGeosErrorMsg](#) (const char *msg)
- Set the current GEOS error message.*
- GAIAGEO_DECLARE void [gaiaSetGeosErrorMsg_r](#) (const void *p_cache, const char *msg)
- Set the current GEOS error message.*
- GAIAGEO_DECLARE void [gaiaSetGeosWarningMsg](#) (const char *msg)
- Set the current GEOS warning message.*
- GAIAGEO_DECLARE void [gaiaSetGeosWarningMsg_r](#) (const void *p_cache, const char *msg)
- Set the current GEOS warning message.*
- GAIAGEO_DECLARE void [gaiaSetGeosAuxErrorMsg](#) (const char *msg)
- Set the current GEOS (auxiliary) error message.*
- GAIAGEO_DECLARE void [gaiaSetGeosAuxErrorMsg_r](#) (const void *p_cache, const char *msg)
- Set the current GEOS (auxiliary) error message.*
- GAIAGEO_DECLARE void * [gaiaToGeos](#) (const [gaiaGeomCollPtr](#) gaia)
- Converts a Geometry object into a GEOS Geometry.*
- GAIAGEO_DECLARE void * [gaiaToGeos_r](#) (const void *p_cache, const [gaiaGeomCollPtr](#) gaia)
- Converts a Geometry object into a GEOS Geometry.*
- GAIAGEO_DECLARE void * [gaiaToGeosSelective](#) (const [gaiaGeomCollPtr](#) gaia, int mode)
- Converts a Geometry object into a GEOS Geometry.*
- GAIAGEO_DECLARE void * [gaiaToGeosSelective_r](#) (const void *p_cache, const [gaiaGeomCollPtr](#) gaia, int mode)
- Converts a Geometry object into a GEOS Geometry.*
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaFromGeos_XY](#) (const void *geos)
- Converts a GEOS Geometry into a Geometry object [XY dims].*
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaFromGeos_XY_r](#) (const void *p_cache, const void *geos)
- Converts a GEOS Geometry into a Geometry object [XY dims].*
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaFromGeos_XYZ](#) (const void *geos)
- Converts a GEOS Geometry into a Geometry object [XYZ dims].*
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaFromGeos_XYZ_r](#) (const void *p_cache, const void *geos)
- Converts a GEOS Geometry into a Geometry object [XYZ dims].*
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaFromGeos_XYM](#) (const void *geos)
- Converts a GEOS Geometry into a Geometry object [XYM dims].*
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaFromGeos_XYM_r](#) (const void *p_cache, const void *geos)

- Converts a GEOS Geometry into a Geometry object [XYM dims].*
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaFromGeos_XYZM](#) (const void *geos)
- Converts a GEOS Geometry into a Geometry object [XYZM dims].*
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaFromGeos_XYZM_r](#) (const void *p_cache, const void *geos)
- Converts a GEOS Geometry into a Geometry object [XYZM dims].*
- GAIAGEO_DECLARE int [gaialsSimple](#) ([gaiaGeomCollPtr](#) geom)
- Checks if a Geometry object represents an OGC Simple Geometry.*
- GAIAGEO_DECLARE int [gaialsSimple_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom)
- Checks if a Geometry object represents an OGC Simple Geometry.*
- GAIAGEO_DECLARE int [gaialsClosed](#) ([gaiaLinestringPtr](#) line)
- Checks if a Linestring object represents an OGC Closed Geometry.*
- GAIAGEO_DECLARE int [gaialsClosedGeom](#) ([gaiaGeomCollPtr](#) geom)
- Checks if a Geometry object represents an OGC Closed Linestring.*
- GAIAGEO_DECLARE int [gaialsClosedGeom_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom)
- Checks if a Geometry object represents an OGC Closed Linestring.*
- GAIAGEO_DECLARE int [gaialsRing](#) ([gaiaLinestringPtr](#) line)
- Checks if a Linestring object represents an OGC Ring Geometry.*
- GAIAGEO_DECLARE int [gaialsRing_r](#) (const void *p_cache, [gaiaLinestringPtr](#) line)
- Checks if a Linestring object represents an OGC Ring Geometry.*
- GAIAGEO_DECLARE int [gaialsValid](#) ([gaiaGeomCollPtr](#) geom)
- Checks if a Geometry object represents an OGC Valid Geometry.*
- GAIAGEO_DECLARE char * [gaialsValidReason](#) ([gaiaGeomCollPtr](#) geom)
- return a TEXT string stating if a Geometry is valid and if not valid, a reason why*
- GAIAGEO_DECLARE char * [gaialsValidReason_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom)
- return a TEXT string stating if a Geometry is valid and if not valid, a reason why*
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaialsValidDetail](#) ([gaiaGeomCollPtr](#) geom)
- return a Geometry detail causing a Geometry to be invalid*
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaialsValidDetailEx](#) ([gaiaGeomCollPtr](#) geom, int esri_flag)
- return a Geometry detail causing a Geometry to be invalid*
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaialsValidDetail_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom)
- return a Geometry detail causing a Geometry to be invalid*
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaialsValidDetailEx_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, int esri_flag)
- return a Geometry detail causing a Geometry to be invalid*
- GAIAGEO_DECLARE int [gaialsValid_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom)
- Checks if a Geometry object represents an OGC Valid Geometry.*
- GAIAGEO_DECLARE int [gaiaGeomCollLength](#) ([gaiaGeomCollPtr](#) geom, double *length)
- Measures the total Length for a Geometry object.*
- GAIAGEO_DECLARE int [gaiaGeomCollLength_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, double *length)
- Measures the total Length for a Geometry object.*
- GAIAGEO_DECLARE int [gaiaGeomCollLengthOrPerimeter](#) ([gaiaGeomCollPtr](#) geom, int perimeter, double *length)
- Measures the total Length or Perimeter for a Geometry object.*
- GAIAGEO_DECLARE int [gaiaGeomCollLengthOrPerimeter_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, int perimeter, double *length)
- Measures the total Length or Perimeter for a Geometry object.*
- GAIAGEO_DECLARE int [gaiaGeomCollArea](#) ([gaiaGeomCollPtr](#) geom, double *area)
- Measures the total Area for a Geometry object.*
- GAIAGEO_DECLARE int [gaiaGeomCollArea_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, double *area)
- Measures the total Area for a Geometry object.*

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaPolygonize](#) ([gaiaGeomCollPtr](#) geom, int force_multi)
Attempts to rearrange a generic Geometry object into a Polygon or MultiPolygon.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaPolygonize_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, int force_multi)
Attempts to rearrange a generic Geometry object into a Polygon or MultiPolygon.
- GAIAGEO_DECLARE int [gaiaGeomCollEquals](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial relationship evaluation: Equals.
- GAIAGEO_DECLARE int [gaiaGeomCollEquals_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial relationship evaluation: Equals.
- GAIAGEO_DECLARE int [gaiaGeomCollDisjoint](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial relationship evaluation: Disjoint.
- GAIAGEO_DECLARE int [gaiaGeomCollDisjoint_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial relationship evaluation: Disjoint.
- GAIAGEO_DECLARE int [gaiaGeomCollPreparedDisjoint](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, unsigned char *blob1, int size1, [gaiaGeomCollPtr](#) geom2, unsigned char *blob2, int size2)
Spatial relationship evaluation: Disjoint (GEOSPreparedGeometry)
- GAIAGEO_DECLARE int [gaiaGeomCollIntersects](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial relationship evaluation: Intesects.
- GAIAGEO_DECLARE int [gaiaGeomCollIntersects_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial relationship evaluation: Intersects.
- GAIAGEO_DECLARE int [gaiaGeomCollPreparedIntersects](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, unsigned char *blob1, int size1, [gaiaGeomCollPtr](#) geom2, unsigned char *blob2, int size2)
Spatial relationship evaluation: Intersects (GEOSPreparedGeometry)
- GAIAGEO_DECLARE int [gaiaGeomCollOverlaps](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial relationship evaluation: Overlaps.
- GAIAGEO_DECLARE int [gaiaGeomCollOverlaps_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial relationship evaluation: Overlaps.
- GAIAGEO_DECLARE int [gaiaGeomCollPreparedOverlaps](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, unsigned char *blob1, int size1, [gaiaGeomCollPtr](#) geom2, unsigned char *blob2, int size2)
Spatial relationship evaluation: Overlaps (GEOSPreparedGeometry)
- GAIAGEO_DECLARE int [gaiaGeomCollCrosses](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial relationship evaluation: Crosses.
- GAIAGEO_DECLARE int [gaiaGeomCollCrosses_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial relationship evaluation: Crosses.
- GAIAGEO_DECLARE int [gaiaGeomCollPreparedCrosses](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, unsigned char *blob1, int size1, [gaiaGeomCollPtr](#) geom2, unsigned char *blob2, int size2)
Spatial relationship evaluation: Crosses (GEOSPreparedGeometry)
- GAIAGEO_DECLARE int [gaiaGeomCollContains](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial relationship evaluation: Contains.
- GAIAGEO_DECLARE int [gaiaGeomCollContains_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial relationship evaluation: Contains.
- GAIAGEO_DECLARE int [gaiaGeomCollPreparedContains](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, unsigned char *blob1, int size1, [gaiaGeomCollPtr](#) geom2, unsigned char *blob2, int size2)
Spatial relationship evaluation: Contains (GEOSPreparedGeometry)
- GAIAGEO_DECLARE int [gaiaGeomCollWithin](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial relationship evaluation: Within.

- GAIAGEO_DECLARE int [gaiaGeomCollWithin_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial relationship evaluation: Within.
- GAIAGEO_DECLARE int [gaiaGeomCollPreparedWithin](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, unsigned char *blob1, int size1, [gaiaGeomCollPtr](#) geom2, unsigned char *blob2, int size2)
Spatial relationship evaluation: Within (GEOSPreparedGeometry)
- GAIAGEO_DECLARE int [gaiaGeomCollTouches](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial relationship evaluation: Touches.
- GAIAGEO_DECLARE int [gaiaGeomCollTouches_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial relationship evaluation: Touches.
- GAIAGEO_DECLARE int [gaiaGeomCollPreparedTouches](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, unsigned char *blob1, int size1, [gaiaGeomCollPtr](#) geom2, unsigned char *blob2, int size2)
Spatial relationship evaluation: Touches (GEOSPreparedGeometry)
- GAIAGEO_DECLARE int [gaiaGeomCollRelate](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, const char *pattern)
Spatial relationship evaluation: Relate.
- GAIAGEO_DECLARE int [gaiaGeomCollRelate_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, const char *pattern)
Spatial relationship evaluation: Relate.
- GAIAGEO_DECLARE char * [gaiaGeomCollRelateBoundaryNodeRule](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, int mode)
Spatial relationship evaluation: Relate Boundary Node Rule.
- GAIAGEO_DECLARE char * [gaiaGeomCollRelateBoundaryNodeRule_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, int mode)
Spatial relationship evaluation: Relate Boundary Node Rule.
- GAIAGEO_DECLARE int [gaiaIntersectionMatrixPatternMatch](#) (const char *matrix, const char *pattern)
Spatial relationship evaluation: comparing two intersection matrices.
- GAIAGEO_DECLARE int [gaiaIntersectionMatrixPatternMatch_r](#) (const void *p_cache, const char *matrix, const char *pattern)
Spatial relationship evaluation: comparing two intersection matrices.
- GAIAGEO_DECLARE int [gaiaGeomCollDistance](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, double *dist)
Calculates the maximum distance intercurring between two Geometry objects.
- GAIAGEO_DECLARE int [gaiaGeomCollDistance_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, double *dist)
Calculates the maximum distance intercurring between two Geometry objects.
- GAIAGEO_DECLARE int [gaiaGeomCollPreparedDistance](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, unsigned char *blob1, int size1, [gaiaGeomCollPtr](#) geom2, unsigned char *blob2, int size2, double *dist)
Spatial relationship evaluation: Distance (GEOSPreparedGeometry)
- GAIAGEO_DECLARE int [gaiaGeomCollPreparedDistanceWithin](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, unsigned char *blob1, int size1, [gaiaGeomCollPtr](#) geom2, unsigned char *blob2, int size2, double dist)
Spatial operator: DistanceWithin (GEOSPreparedGeometry)
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaGeometryIntersection](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial operator: Intersection.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaGeometryIntersection_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial operator: Intersection.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaGeometryUnion](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial operator: Union.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaGeometryUnion_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial operator: Union.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaUnionCascaded](#) ([gaiaGeomCollPtr](#) geom)
Spatial operator: Union Cascaded.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaUnionCascaded_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom)
Spatial operator: Union Cascaded.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaGeometryDifference](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial operator: Difference.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaGeometryDifference_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial operator: Difference.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaGeometrySymDifference](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial operator: SymDifference.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaGeometrySymDifference_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial operator: SymDifference.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaBoundary](#) ([gaiaGeomCollPtr](#) geom)
Spatial operator: Boundary.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaBoundary_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom)
Spatial operator: Boundary.
- GAIAGEO_DECLARE int [gaiaGeomCollCentroid](#) ([gaiaGeomCollPtr](#) geom, double *x, double *y)
Spatial operator: Centroid.
- GAIAGEO_DECLARE int [gaiaGeomCollCentroid_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, double *x, double *y)
Spatial operator: Centroid.
- GAIAGEO_DECLARE int [gaiaGetPointOnSurface](#) ([gaiaGeomCollPtr](#) geom, double *x, double *y)
Spatial operator: PointOnSurface.
- GAIAGEO_DECLARE int [gaiaGetPointOnSurface_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, double *x, double *y)
Spatial operator: PointOnSurface.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaGeomCollSimplify](#) ([gaiaGeomCollPtr](#) geom, double tolerance)
Spatial operator: Simplify.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaGeomCollSimplify_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, double tolerance)
Spatial operator: Simplify.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaGeomCollSimplifyPreserveTopology](#) ([gaiaGeomCollPtr](#) geom, double tolerance)
Spatial operator: Simplify [preserving topology].
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaGeomCollSimplifyPreserveTopology_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, double tolerance)
Spatial operator: Simplify [preserving topology].
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaConvexHull](#) ([gaiaGeomCollPtr](#) geom)
Spatial operator: ConvexHull.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaConvexHull_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom)
Spatial operator: ConvexHull.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaGeomCollBuffer](#) ([gaiaGeomCollPtr](#) geom, double radius, int points)
Spatial operator: Buffer.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaGeomCollBuffer_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, double radius, int points)
Spatial operator: Buffer.
- GAIAGEO_DECLARE int [gaiaHausdorffDistance](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, double *dist)
Calculates the Hausdorff distance intercurring between two Geometry objects.
- GAIAGEO_DECLARE int [gaiaHausdorffDistance_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, double *dist)
Calculates the Hausdorff distance intercurring between two Geometry objects.
- GAIAGEO_DECLARE int [gaiaHausdorffDistanceDensify](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, double densify_fract, double *dist)
Calculates the Hausdorff distance intercurring between two Geometry objects.
- GAIAGEO_DECLARE int [gaiaHausdorffDistanceDensify_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, double densify_fract, double *dist)
Calculates the Hausdorff distance intercurring between two Geometry objects.
- GAIAGEO_DECLARE int [gaiaFrechetDistance](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, double *dist)
Calculates the Frechet distance intercurring between two Geometry objects.
- GAIAGEO_DECLARE int [gaiaFrechetDistance_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, double *dist)
Calculates the Frechet distance intercurring between two Geometry objects.
- GAIAGEO_DECLARE int [gaiaFrechetDistanceDensify](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, double densify_fract, double *dist)
Calculates the Frechet distance intercurring between two Geometry objects.
- GAIAGEO_DECLARE int [gaiaFrechetDistanceDensify_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, double densify_fract, double *dist)
Calculates the Frechet distance intercurring between two Geometry objects.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaMinimumRotatedRectangle](#) ([gaiaGeomCollPtr](#) geom)
Calculates the minimum rotated rectangular POLYGON which encloses the input geometry.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaMinimumRotatedRectangle_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom)
Calculates the minimum rotated rectangular POLYGON which encloses the input geometry.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaMaximumInscribedCircle](#) ([gaiaGeomCollPtr](#) geom, double tolerance)
Constructs the Maximum Inscribed Circle for a polygonal geometry, up to a specified tolerance.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaMaximumInscribedCircle_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, double tolerance)
Constructs the Maximum Inscribed Circle for a polygonal geometry, up to a specified tolerance.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaMinimumBoundingCircle](#) ([gaiaGeomCollPtr](#) geom, double *radius, [gaiaGeomCollPtr](#) *center)
Constructs the Minimum Bounding Circle for a generic geometry.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaMinimumBoundingCircle_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, double *radius, [gaiaGeomCollPtr](#) *center)
Constructs the Minimum Bounding Circle for a generic geometry.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaLargestEmptyCircle](#) ([gaiaGeomCollPtr](#) geom, [gaiaGeomCollPtr](#) boundary, double tolerance)
Constructs the Largest Empty Circle for a set of obstacle geometries, up to a specified tolerance.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaLargestEmptyCircle_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, [gaiaGeomCollPtr](#) boundary, double tolerance)
Constructs the Largest Empty Circle for a set of obstacle geometries, up to a specified tolerance.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaMinimumWidth](#) ([gaiaGeomCollPtr](#) geom)
Returns a LINESTRING geometry which represents the minimum diameter of the geometry.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaMinimumWidth_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom)
Returns a LINESTRING geometry which represents the minimum diameter of the geometry.
- GAIAGEO_DECLARE int [gaiaMinimumClearance](#) ([gaiaGeomCollPtr](#) geom, double *clearance)
Computes the minimum clearance of a geometry.
- GAIAGEO_DECLARE int [gaiaMinimumClearance_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, double *clearance)
Computes the minimum clearance of a geometry.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaMinimumClearanceLine](#) ([gaiaGeomCollPtr](#) geom)
Returns a LINESTRING geometry which represents the minimum clearance of a geometry.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaMinimumClearanceLine_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom)
Returns a LINESTRING geometry which represents the minimum clearance of a geometry.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaOffsetCurve](#) ([gaiaGeomCollPtr](#) geom, double radius, int points, int left_right)
Spatial operator: Offset Curve.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaOffsetCurve_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, double radius, int points, int left_right)
Spatial operator: Offset Curve.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaSingleSidedBuffer](#) ([gaiaGeomCollPtr](#) geom, double radius, int points, int left_right)
Spatial operator: Single Sided Buffer.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaSingleSidedBuffer_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, double radius, int points, int left_right)
Spatial operator: Single Sided Buffer.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaSharedPaths](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial operator: Shared Paths.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaSharedPaths_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial operator: Shared Paths.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaLineInterpolatePoint](#) ([gaiaGeomCollPtr](#) In_geom, double fraction)
Spatial operator: Line Interpolate Point.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaLineInterpolatePoint_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) In_geom, double fraction)
Spatial operator: Line Interpolate Point.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaLineInterpolateEquidistantPoints](#) ([gaiaGeomCollPtr](#) In_geom, double distance)
Spatial operator: Line Interpolate Equidistant Points.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaLineInterpolateEquidistantPoints_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) In_geom, double distance)
Spatial operator: Line Interpolate Equidistant Points.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaLineSubstring](#) ([gaiaGeomCollPtr](#) In_geom, double start_fraction, double end_fraction)
Spatial operator: Line Substring.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaLineSubstring_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) In_geom, double start_fraction, double end_fraction)
Spatial operator: Line Substring.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaShortestLine](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial operator: Shortest Line.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaShortestLine_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial operator: Shortest Line.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaSnap](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, double tolerance)
Spatial operator: Snap.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaSnap_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, double tolerance)
Spatial operator: Snap.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaLineMerge](#) ([gaiaGeomCollPtr](#) geom)
Spatial operator: Line Merge.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaLineMerge_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom)
Spatial operator: Line Merge.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaLinesCutAtNodes](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Spatial operator: Line Cut At Nodes.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaUnaryUnion](#) ([gaiaGeomCollPtr](#) geom)
Spatial operator: Unary Union.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaUnaryUnion_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom)
Spatial operator: Unary Union.
- GAIAGEO_DECLARE double [gaiaLineLocatePoint](#) ([gaiaGeomCollPtr](#) ln_geom, [gaiaGeomCollPtr](#) pt_geom)
Determines the location of the closest Point on Linestring to the given Point.
- GAIAGEO_DECLARE double [gaiaLineLocatePoint_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) ln_geom, [gaiaGeomCollPtr](#) pt_geom)
Determines the location of the closest Point on Linestring to the given Point.
- GAIAGEO_DECLARE int [gaiaGeomCollCovers](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Topology check: test if a Geometry covers another one.
- GAIAGEO_DECLARE int [gaiaGeomCollCovers_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Topology check: test if a Geometry covers another one.
- GAIAGEO_DECLARE int [gaiaGeomCollPreparedCovers](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, unsigned char *blob1, int size1, [gaiaGeomCollPtr](#) geom2, unsigned char *blob2, int size2)
Topology check: test if a Geometry covers another one (GEOSPreparedGeometry)
- GAIAGEO_DECLARE int [gaiaGeomCollCoveredBy](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Topology check: test if a Geometry is covered by another one.
- GAIAGEO_DECLARE int [gaiaGeomCollCoveredBy_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
Topology check: test if a Geometry is covered by another one.
- GAIAGEO_DECLARE int [gaiaGeomCollPreparedCoveredBy](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, unsigned char *blob1, int size1, [gaiaGeomCollPtr](#) geom2, unsigned char *blob2, int size2)
Topology check: test if a Geometry is covered by another one (GEOSPreparedGeometry)
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaSquareGrid](#) ([gaiaGeomCollPtr](#) geom, double origin_x, double origin_y, double size, int mode)
Utility function: SquareGrid.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaSquareGrid_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, double origin_x, double origin_y, double size, int mode)
Utility function: SquareGrid.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaTriangularGrid](#) ([gaiaGeomCollPtr](#) geom, double origin_x, double origin_y, double size, int mode)
Utility function: TriangularGrid.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaTriangularGrid_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, double origin_x, double origin_y, double size, int mode)

Utility function: TriangularGrid.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaHexagonalGrid](#) ([gaiaGeomCollPtr](#) geom, double origin_x, double origin_y, double size, int mode)

Utility function: HexagonalGrid.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaHexagonalGrid_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, double origin_x, double origin_y, double size, int mode)

Utility function: HexagonalGrid.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaGeosDensify](#) ([gaiaGeomCollPtr](#) geom, double tolerance)

Densifies a geometry using a given distance tolerance.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaGeosDensify_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, double tolerance)

Densifies a geometry using a given distance tolerance.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaConstrainedDelaunayTriangulation](#) ([gaiaGeomCollPtr](#) geom)

Constrained Delaunay Triangulation.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaConstrainedDelaunayTriangulation_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom)

Constrained Delaunay Triangulation.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaGeosMakeValid](#) ([gaiaGeomCollPtr](#) geom, int keep_discarded)

Utility function: GeosMakeValid.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaGeosMakeValid_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, int keep_discarded)

Utility function: GeosMakeValid.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaReducePrecision](#) ([gaiaGeomCollPtr](#) geom, double grid_size)

Change the coordinate precision of a geometry.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaReducePrecision_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, double grid_size)

Change the coordinate precision of a geometry.

- GAIAGEO_DECLARE int [gaiaHilbertCode](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, int level, unsigned int *code)

Hilbert Code.

- GAIAGEO_DECLARE int [gaiaHilbertCode_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, int level, unsigned int *code)

Hilbert Code.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaGeosConcaveHull](#) ([gaiaGeomCollPtr](#) geom, double ratio, int allow_holes)

Concave Hull (the GEOS way)

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaGeosConcaveHull_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, double ratio, int allow_holes)

Concave Hull (the GEOS way)

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaDelaunayTriangulation](#) ([gaiaGeomCollPtr](#) geom, double tolerance, int only_edges)

Delaunay Triangulation.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaDelaunayTriangulation_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, double tolerance, int only_edges)

Delaunay Triangulation.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaVoronoiDiagram](#) ([gaiaGeomCollPtr](#) geom, double extra_frame_size, double tolerance, int only_edges)

Voronoi Diagram.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaVoronoiDiagram_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, double extra_frame_size, double tolerance, int only_edges)

Voronoi Diagram.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaConcaveHull](#) ([gaiaGeomCollPtr](#) geom, double factor, double tolerance, int allow_holes)

- Concave Hull.*
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaConcaveHull_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, double factor, double tolerance, int allow_holes)
- Concave Hull.*
- GAIAGEO_DECLARE void [gaiaResetRtTopoMsg](#) (const void *p_cache)
- Resets the RTTOPO error and warning messages to an empty state.*
- GAIAGEO_DECLARE const char * [gaiaGetRtTopoErrorMsg](#) (const void *p_cache)
- Return the latest RTTOPO error message (if any)*
- GAIAGEO_DECLARE const char * [gaiaGetRtTopoWarningMsg](#) (const void *p_cache)
- Return the latest RTTOPO warning message (if any)*
- GAIAGEO_DECLARE void [gaiaSetRtTopoErrorMsg](#) (const void *p_cache, const char *msg)
- Set the current RTTOPO error message.*
- GAIAGEO_DECLARE void [gaiaSetRtTopoWarningMsg](#) (const void *p_cache, const char *msg)
- Set the current RTTOPO warning message.*
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaMakeValid](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom)
- Utility function: MakeValid.*
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaMakeValidDiscarded](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom)
- Utility function: MakeValidDiscarded.*
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaSegmentize](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, double dist)
- Utility function: Segmentize.*
- GAIAGEO_DECLARE int [gaiaAzimuth](#) (const void *p_cache, double xa, double ya, double xb, double yb, double *azimuth)
- Utility function: Azimuth.*
- GAIAGEO_DECLARE int [gaiaEllipsoidAzimuth](#) (const void *p_cache, double xa, double ya, double xb, double yb, double a, double b, double *azimuth)
- Utility function: EllipsoidAzimuth.*
- GAIAGEO_DECLARE int [gaiaProjectedPoint](#) (const void *p_cache, double x1, double y1, double a, double b, double distance, double azimuth, double *x2, double *y2)
- Utility function: ProjectedPoint.*
- GAIAGEO_DECLARE char * [gaiaGeoHash](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, int precision)
- Utility function: GeoHash.*
- GAIAGEO_DECLARE char * [gaiaAsX3D](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, const char *srs, int precision, int options, const char *refid)
- Utility function: AsX3D.*
- GAIAGEO_DECLARE int [gaia3DDistance](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, double *dist)
- Calculates the minimum 3D distance intercurring between two Geometry objects.*
- GAIAGEO_DECLARE int [gaiaMaxDistance](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, double *dist)
- Calculates the maximum 2D distance intercurring between two Geometry objects.*
- GAIAGEO_DECLARE int [gaia3DMaxDistance](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, double *dist)
- Calculates the maximum 3D distance intercurring between two Geometry objects.*
- GAIAGEO_DECLARE int [gaia3dLength](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, double *length)
- Calculates the 2D or 3D Length for a Linestring or Multilinestring accordingly to the dimensions of Geometry.*
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaSplit](#) (const void *p_cache, [gaiaGeomCollPtr](#) input, [gaiaGeomCollPtr](#) blade)
- Utility function: Split.*
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaSplitLeft](#) (const void *p_cache, [gaiaGeomCollPtr](#) input, [gaiaGeomCollPtr](#) blade)

Utility function: SplitLeft.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaSplitRight](#) (const void *p_cache, [gaiaGeomCollPtr](#) input, [gaiaGeomCollPtr](#) blade)

Utility function: SplitRight.

- GAIAGEO_DECLARE int [gaiaGeodesicArea](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, double a, double b, int use_ellipsoid, double *area)

Measures the total Area for a Geometry object (geodesic)

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaNodeLines](#) (const void *p_cache, [gaiaGeomCollPtr](#) input)

Utility function: re-noding lines.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaSubdivide](#) (const void *p_cache, [gaiaGeomCollPtr](#) input, int max_vertices)

Utility function: subdividing Geometry.

- GAIAGEO_DECLARE int [gaiaToTWKB](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, unsigned char precision_xy, unsigned char precision_z, unsigned char precision_m, int with_size, int with_bbox, unsigned char **twkb, int *size_twkb)

Converts a native binary Geometry into a compressed TWKB Geometry.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaFromTWKB](#) (const void *p_cache, const unsigned char *twkb, int twkb_size, int srid)

Converts a compressed TWKB Geometry into a native binary Geometry.

- GAIAGEO_DECLARE int [gaiaAsEncodedPolyLine](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom, unsigned char precision, char **encoded, int *len)

Converts a native binary Geometry into a GoogleMaps encoded PolyLine.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaLineFromEncodedPolyline](#) (const void *p_cache, const char *encoded, unsigned char precision)

Converts a GoogleMaps encoded PolyLine into a native binary Geometry (Linestring)

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaDrapeLine](#) (sqlite3 *db_handle, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, double tolerance)

Utility function: DrapeLine.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaDrapeLineExceptions](#) (sqlite3 *db_handle, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, double tolerance, int interpolated)

Utility function: DrapeLineExceptions.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaSnapToGrid](#) ([gaiaGeomCollPtr](#) geom, double origin_x, double origin_y, double origin_z, double origin_m, double size_x, double size_y, double size_z, double size_m)

Utility function: SnapToGrid.

6.5.1 Detailed Description

Geometry handling functions: advanced.

6.5.2 Function Documentation

6.5.2.1 [gaia3DDistance\(\)](#)

```
GAIAGEO_DECLARE int gaia3DDistance (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2,
    double * dist )
```

Calculates the minimum 3D distance intercurring between two Geometry objects.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object
<i>geom2</i>	the second Geometry object
<i>dist</i>	on completion this variable will contain the calculated distance

Returns

0 on failure: any other value on success.

See also

[gaiaGeomCollDistance](#), [gaiaMaxDistance](#), [gaia3DMaxDisance](#)

Note

this function computes the 3D cartesian distance (if Z is supported)

Remarks

RTTOPO support required.

6.5.2.2 [gaia3dLength\(\)](#)

```
GAIA GEO_DECLARE int gaia3dLength (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    double * length )
```

Calculates the 2D or 3D Length for a Linestring or Multilinestring accordingly to the dimensions of Geometry.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	the Geometry object
<i>length</i>	on completion this variable will contain the calculated length

Returns

0 on failure: any other value on success.

See also

[gaiaGeomCollDistance](#)

Remarks

RTTOPO support required.

6.5.2.3 gaia3DMaxDistance()

```
GAIA GEO_DECLARE int gaia3DMaxDistance (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2,
    double * dist )
```

Calculates the maximum 3D distance intercurring between two Geometry objects.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object
<i>geom2</i>	the second Geometry object
<i>dist</i>	on completion this variable will contain the calculated distance

Returns

0 on failure: any other value on success.

See also

[gaiaGeomCollDistance](#), [gaia3DDistance](#), [gaiaMaxDistance](#)

Note

this function computes the 3D maximum cartesian distance (if Z is supported)

Remarks

RTTOPO support required.

6.5.2.4 gaiaAsEncodedPolyLine()

```
GAIA GEO_DECLARE int gaiaAsEncodedPolyLine (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    unsigned char precision,
    char ** encoded,
    int * len )
```

Converts a native binary Geometry into a GoogleMaps encoded PolyLine.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	the Geometry to be converted
<i>precision</i>	how much precision (decimal digits)
<i>encoded</i>	on succesfull completion this pointer will reference the GoogleMaps encoded PolyLine
<i>len</i>	on succesfull completion this pointer will reference the length (in bytes) of the GoogleMaps encoded PolyLine

Returns

0 on failure: any other value on success.

See also

[gaiaLineFromEncodedPolyline](#)

Note

you are responsible to free (before or after) the TWKB geometry created by [gaiaToTWKB\(\)](#).

Remarks

RTTOPO support required.

6.5.2.5 **gaiaAsX3D()**

```
GAIA GEO_DECLARE char* gaiaAsX3D (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    const char * srs,
    int precision,
    int options,
    const char * refid )
```

Utility function: AsX3D.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	the input geometry.
<i>srs</i>	the WKT SRS definition.
<i>precision</i>	the expected precision (coord decimal digits).
<i>options</i>	
<i>refid</i>	the X3D namespace

Returns

NULL on failure: a null-terminated text string on success

Note

you are responsible to free (before or after) any text string returned by [gaiaAsX3D\(\)](#)

Remarks

RTTOPO support required.

6.5.2.6 gaiaAzimuth()

```
GAIAGEO_DECLARE int gaiaAzimuth (
    const void * p_cache,
    double xa,
    double ya,
    double xb,
    double yb,
    double * azimuth )
```

Utility function: Azimuth.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>xa</i>	the X coordinate of PointA.
<i>ya</i>	the Y coordinate of PointA.
<i>xb</i>	the X ccoordinate of PointB.
<i>yb</i>	the Y coordinate of PointB.
<i>azimuth</i>	on completion this variable will contain the angle in radians from the horizontal of the vector defined by pointA and pointB. Angle is computed clockwise from down-to-up: on the clock: 12=0; 3=PI/2; 6=PI; 9=3PI/2.

Returns

0 on failure: any other value on success

See also

[gaiaProjectedPoint](#)

Remarks

RTTOPO support required.

6.5.2.7 `gaiaBoundary()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaBoundary (
    gaiaGeomCollPtr geom )
```

Spatial operator: Boundary.

Parameters

<code>geom</code>	the Geometry object to be evaluated
-------------------	-------------------------------------

Returns

the pointer to newly created Geometry object representing the geometry Boundary of the input Geometry:
NULL on failure.

See also

`gaiaBoudary_r`, [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaBoundary\(\)](#)
not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.8 `gaiaBoundary_r()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaBoundary_r (
    const void * p_cache,
    gaiaGeomCollPtr geom )
```

Spatial operator: Boundary.

Parameters

<code>p_cache</code>	a memory pointer returned by spatialite_alloc_connection()
<code>geom</code>	the Geometry object to be evaluated

Returns

the pointer to newly created Geometry object representing the geometry Boundary of the input Geometry:
NULL on failure.

See also

[gaiaBoudary](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaBoundary_r\(\)](#) reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.9 gaiaConcaveHull()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaConcaveHull (
    gaiaGeomCollPtr geom,
    double factor,
    double tolerance,
    int allow_holes )
```

Concave Hull.

Parameters

<i>geom</i>	pointer to input Geometry object.
<i>factor</i>	multiplier used for filtering Delaunay triangles: please read the note.
<i>tolerance</i>	optional snapping tolerance.
<i>allow_holes</i>	if FALSE any interior hole will be suppressed.

Returns

the pointer to newly created Geometry object (always of the Polygon type): NULL on failure.
NULL will be returned if any argument is invalid.

See also

[gaiaConcaveHull_r](#), [gaiaFreeGeomColl](#), [gaiaDelaunayTriangulation](#), [gaiaGeosConcaveHull](#), [gaiaGeosConcaveHull_r](#)

Note

This function will first create the Delaunay Triangulation corresponding to input Geometry, determining at the same time the **standard deviation** for all edge's lengths.
Then in a second pass all Delaunay's triangles will be filtered, and all triangles presenting at least one edge longer than **standard deviation * factor** will be discarded.
All filtered triangles will then be merged altogether so to create the Concave Hull.

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaConcaveHull\(\)](#)
not reentrant and thread unsafe.

Remarks

GEOS-TRUNK support required.

6.5.2.10 `gaiaConcaveHull_r()`

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaConcaveHull_r (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    double factor,
    double tolerance,
    int allow_holes )
```

Concave Hull.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to input Geometry object.
<i>factor</i>	multiplier used for filtering Delaunay triangles: please read the note.
<i>tolerance</i>	optional snapping tolerance.
<i>allow_holes</i>	if FALSE any interior hole will be suppressed.

Returns

the pointer to newly created Geometry object (always of the Polygon type): NULL on failure.
 NULL will be returned if any argument is invalid.

See also

[gaiaConcaveHull](#), [gaiaFreeGeomColl](#), [gaiaDelaunayTriangulation](#), [gaiaGeosConcaveHull](#), [gaiaGeosConcaveHull_r](#)

Note

This function will first create the Delaunay Triangulation corresponding to input Geometry, determining at the same time the **standard deviation** for all edge's lengths.
 Then in a second pass all Delaunay's triangles will be filtered, and all triangles presenting at least one edge longer than **standard deviation * factor** will be discarded.
 All filtered triangles will then be merged altogether so to create the Concave Hull.

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaConcaveHull_r\(\)](#)
 reentrant and thread-safe.

Remarks

GEOS-TRUNK support required.

6.5.2.11 `gaiaConstrainedDelaunayTriangulation()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaConstrainedDelaunayTriangulation (
    gaiaGeomCollPtr geom )
```

Constrained Delaunay Triangulation.

Parameters

<i>geom</i>	pointer to input Geometry object.
-------------	-----------------------------------

Returns

the pointer to newly created Geometry object: NULL on failure.
 NULL will be returned if any argument is invalid or if the input geometry does not contain any Polygon.

See also

[gaiaConstrainedDelaunayTriangulation_r](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaConstrainedDelaunayTriangulation\(\)](#)
 not reentrant and thread unsafe.

Remarks

GEOS_3100 support required.

6.5.2.12 `gaiaConstrainedDelaunayTriangulation_r()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaConstrainedDelaunayTriangulation_r (
    const void * p_cache,
    gaiaGeomCollPtr geom )
```

Constrained Delaunay Triangulation.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to input Geometry object.

Returns

the pointer to newly created Geometry object: NULL on failure.
 NULL will be returned if any argument is invalid or if the input geometry does not contain any Polygon.

See also

[gaiaConstrainedDelaunayTriangulation](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaConstrainedDelaunayTriangulation_r\(\)](#) reentrant and thread-safe.

Remarks

GEOS_3100 support required.

6.5.2.13 [gaiaConvexHull\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaConvexHull (
    gaiaGeomCollPtr geom )
```

Spatial operator: ConvexHull.

Parameters

<i>geom</i>	the input Geometry object
-------------	---------------------------

Returns

the pointer to newly created Geometry object representing the ConvexHull of input Geometry: NULL on failure.

See also

[gaiaConvexHull_r](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaConvexHull\(\)](#) not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.14 [gaiaConvexHull_r\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaConvexHull\_r (
    const void * p_cache,
    gaiaGeomCollPtr geom )
```

Spatial operator: ConvexHull.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	the input Geometry object

Returns

the pointer to newly created Geometry object representing the ConvexHull of input Geometry: NULL on failure.

See also

[gaiaConvexHull](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaConvexHull_r\(\)](#) reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.15 [gaiaCriticalPointFromGEOSmsg\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaCriticalPointFromGEOSmsg (
    void )
```

Attempts to (possible) return a Point Geometry extracted from the latest GEOS error / warning message.

Returns

a Point Geometry: NULL if no warning/error was previously found or if the current GEOS message doesn't contains a critical Point.

See also

[gaiaCriticalPointFromGEOSmsg_r](#), [gaiaResetGeosMsg](#), [gaiaGetGeosErrorMsg](#), [gaiaGetGeosWarningMsg](#), [gaiaSetGeosErrorMsg](#), [gaiaSetGeosWarningMsg](#), [gaiaSetGeosAuxErrorMsg](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.16 [gaiaCriticalPointFromGEOSmsg_r\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaCriticalPointFromGEOSmsg\_r (
    const void * p_cache )
```

Attempts to (possible) return a Point Geometry extracted from the latest GEOS error / warning message.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
----------------	--

Returns

a Point Geometry: NULL if no warning/error was previously found or if the current GEOS message doesn't contains a critical Point.

See also

[gaiaCriticalPointFromGEOSmsg_r](#), [gaiaResetGeosMsg](#), [gaiaGetGeosErrorMsg](#), [gaiaGetGeosWarningMsg](#), [gaiaSetGeosErrorMsg](#), [gaiaSetGeosWarningMsg](#), [gaiaSetGeosAuxErrorMsg](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.17 gaiaDegsToRads()

```
GAIA GEO_DECLARE double gaiaDegsToRads (
    double degs )
```

Converts and angle from Degrees into Radians.

Parameters

<i>degs</i>	the angle measured in Degrees.
-------------	--------------------------------

Returns

the angle measured in Radians.

See also

[gaiaRadsToDegs](#)

Remarks

PROJ support required

6.5.2.18 `gaiaDelaunayTriangulation()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaDelaunayTriangulation (
    gaiaGeomCollPtr geom,
    double tolerance,
    int only_edges )
```

Delaunay Triangulation.

Parameters

<i>geom</i>	pointer to input Geometry object.
<i>tolerance</i>	optional snapping tolerance.
<i>only_edges</i>	if non-zero will return a MULTILINESTRING, otherwise it will return a MULTIPOLYGON containing triangular POLYGONS.

Returns

the pointer to newly created Geometry object: NULL on failure.
NULL will be returned if any argument is invalid.

See also

[gaiaDelaunayTriangulation_r](#), [gaiaFreeGeomColl](#), [gaiaVoronoiDiagram](#), [gaiaConcaveHull](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaDelaunayTriangulation\(\)](#)
not reentrant and thread unsafe.

Remarks

GEOS-TRUNK support required.

6.5.2.19 `gaiaDelaunayTriangulation_r()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaDelaunayTriangulation_r (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    double tolerance,
    int only_edges )
```

Delaunay Triangulation.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to input Geometry object.
<i>tolerance</i>	optional snapping tolerance. Generated by Doxygen
<i>only_edges</i>	if non-zero will return a MULTILINESTRING, otherwise it will return a MULTIPOLYGON containing triangular POLYGONS.

Returns

the pointer to newly created Geometry object: NULL on failure.
NULL will be returned if any argument is invalid.

See also

[gaiaDelaunayTriangulation](#), [gaiaFreeGeomColl](#), [gaiaVoronoiDiagram](#), [gaiaConcaveHull](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaDelaunayTriangulation_r\(\)](#) reentrant and thread-safe.

Remarks

GEOS-TRUNK support required.

6.5.2.20 [gaiaDrapeLine\(\)](#)

```
GAIA_GEO_DECLARE gaiaGeomCollPtr gaiaDrapeLine (  
    sqlite3 * db_handle,  
    gaiaGeomCollPtr geom1,  
    gaiaGeomCollPtr geom2,  
    double tolerance )
```

Utility function: [DrapeLine](#).

Parameters

<i>db_handle</i>	pointer to the current DB connection.
<i>geom1</i>	the first Geometry object (expected to be a 2D Linestring).
<i>geom2</i>	the second Geometry object (expected to be a 3D Linestring).
<i>tolerance</i>	tolerance radius.

Returns

the pointer to newly created Geometry object: NULL on failure.
this function will return a Geometry of the Linestring type having all points defined by *geom1* and dimensions as defined by *geom2*. Missing Z and/or M coords will be recovered by corresponding points found in *geom2* within the given tolerance radius.
both *geom1* and *geom2* must share the same SRID.

See also

[gaiaFreeGeomColl](#), [gaiaDrapeLineExceptions](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaDrapeLine\(\)](#)

6.5.2.21 gaiaDrapeLineExceptions()

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaDrapeLineExceptions (
    sqlite3 * db_handle,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2,
    double tolerance,
    int interpolated )
```

Utility function: DrapeLineExceptions.

Parameters

<i>db_handle</i>	pointer to the current DB connection.
<i>geom1</i>	the first Geometry object (expected to be a 2D Linestring).
<i>geom2</i>	the second Geometry object (expected to be a 3D Linestring).
<i>tolerance</i>	tolerance radius.
<i>interpolated</i>	boolean: if TRUE all Vertices for whom Z (and/or M) values had been succesfully interpolated will be considered as valid.

Returns

the pointer to newly created Geometry object: NULL on failure.
 this function will return a Geometry of the MultiPoint type containing all Vertices from geom1 lacking a corresponding Vertex in geom2, thus leading to dubious Z and/or M coords.
 both geom1 and geom2 must share the same SRID.

See also

[gaiaFreeGeomColl](#), [gaiaDrapeLine](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaDrapeLine\(\)](#)

6.5.2.22 gaiaEllipsoidAzimuth()

```
GAIA GEO_DECLARE int gaiaEllipsoidAzimuth (
    const void * p_cache,
    double xa,
```

```
double ya,
double xb,
double yb,
double a,
double b,
double * azimuth )
```

Utility function: EllipsoidAzimuth.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>xa</i>	the X coordinate of PointA.
<i>ya</i>	the Y coordinate of PointA.
<i>xb</i>	the X coordinate of PointB.
<i>yb</i>	the Y coordinate of PointB.
<i>a</i>	major axis of the reference spheroid.
<i>b</i>	minor axis of the reference spheroid.
<i>azimuth</i>	on completion this variable will contain the angle in radians from the horizontal of the vector defined by pointA and pointB. Angle is computed clockwise from down-to-up: on the clock: 12=0; 3=PI/2; 6=PI; 9=3PI/2.

Returns

0 on failure: any other value on success

See also

[gaiaAzimuth](#)

Remarks

RTTOPO support required.

6.5.2.23 [gaiaFrechetDistance\(\)](#)

```
GAIAGEO_DECLARE int gaiaFrechetDistance (
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2,
    double * dist )
```

Calculates the Frechet distance intercurring between two Geometry objects.

Parameters

<i>geom1</i>	pointer to first Geometry object
<i>geom2</i>	pointer to second Geometry object
<i>dist</i>	on completion this variable will contain the calculated Frechet distance

Returns

0 on failure: any other value on success.

See also

[gaiaFrechetDistance_r](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS-ADVANCED support required.

6.5.2.24 gaiaFrechetDistance_r()

```
GAIA GEO_DECLARE int gaiaFrechetDistance_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2,
    double * dist )
```

Calculates the Frechet distance intercurring between two Geometry objects.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	pointer to first Geometry object
<i>geom2</i>	pointer to second Geometry object
<i>dist</i>	on completion this variable will contain the calculated Frechet distance

Returns

0 on failure: any other value on success.

See also

[gaiaFrechetDistance](#)

Note

reentrant and thread-safe.

Remarks

GEOS-ADVANCED support required.

6.5.2.25 `gaiaFrechetDistanceDensify()`

```
GAIA GEO_DECLARE int gaiaFrechetDistanceDensify (
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2,
    double densify_fract,
    double * dist )
```

Calculates the Frechet distance intercurring between two Geometry objects.

Parameters

<i>geom1</i>	pointer to first Geometry object
<i>geom2</i>	pointer to second Geometry object
<i>densify_fract</i>	fraction (in the range 0.0 / 1.0) by which to densify each segment. Each segment will be split into a number of equal-length subsegments, whose fraction of the total length is closest to the given fraction
<i>dist</i>	on completion this variable will contain the calculated Frechet distance

Returns

0 on failure: any other value on success.

See also

[gaiaFrechetDistance_r](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS-ADVANCED support required.

6.5.2.26 `gaiaFrechetDistanceDensify_r()`

```
GAIA GEO_DECLARE int gaiaFrechetDistanceDensify_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2,
    double densify_fract,
    double * dist )
```

Calculates the Frechet distance intercurring between two Geometry objects.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	pointer to first Geometry object
<i>geom2</i>	pointer to second Geometry object
<i>densify_fract</i>	fraction (in the range 0.0 / 1.0) by which to densify each segment. Each segment will be split into a number of equal-length subsegments, whose fraction of the total length is closest to the given fraction
<i>dist</i>	on completion this variable will contain the calculated Frechet distance

Returns

0 on failure: any other value on success.

See also

[gaiaFrechetDistance](#)

Note

reentrant and thread-safe.

Remarks

GEOS-ADVANCED support required.

6.5.2.27 [gaiaFromGeos_XY\(\)](#)

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaFromGeos\_XY (
    const void * geos )
```

Converts a GEOS Geometry into a Geometry object [XY dims].

Parameters

<i>geos</i>	handle to GEOS Geometry
-------------	-------------------------

Returns

the pointer to the newly created Geometry object

See also

[gaiaFromGeos_XY_r](#), [gaiaToGeos](#), [gaiaFromGeos_XYZ](#), [gaiaFromGeos_XYM](#), [gaiaFromGeos_XYZM](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaFromGeos_XY\(\)](#)
not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.28 gaiaFromGeos_XY_r()

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaFromGeos_XY_r (  
    const void * p_cache,  
    const void * geos )
```

Converts a GEOS Geometry into a Geometry object [XY dims].

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geos</i>	handle to GEOS Geometry

Returns

the pointer to the newly created Geometry object

See also

[gaiaFromGeos_XY](#), [gaiaToGeos](#), [gaiaFromGeos_XYZ](#), [gaiaFromGeos_XYM](#), [gaiaFromGeos_XYZM](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaFromGeos_XY_r\(\)](#)
reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.29 gaiaFromGeos_XYM()

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaFromGeos_XYM (  
    const void * geos )
```

Converts a GEOS Geometry into a Geometry object [XYM dims].

Parameters

<i>geos</i>	handle to GEOS Geometry
-------------	-------------------------

Returns

the pointer to the newly created Geometry object

See also

[gaiaFromGeos_XYM_r](#), [gaiaToGeos](#), [gaiaFromGeos_XY](#), [gaiaFromGeos_XYZ](#), [gaiaFromGeos_XYZM](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaFromGeos_XYM\(\)](#)
not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.30 [gaiaFromGeos_XYM_r\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaFromGeos\_XYM\_r (
    const void * p_cache,
    const void * geos )
```

Converts a GEOS Geometry into a Geometry object [XYM dims].

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geos</i>	handle to GEOS Geometry

Returns

the pointer to the newly created Geometry object

See also

[gaiaFromGeos_XYM](#), [gaiaToGeos](#), [gaiaFromGeos_XY](#), [gaiaFromGeos_XYZ](#), [gaiaFromGeos_XYZM](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaFromGeos_XYM_r\(\)](#) reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.31 gaiaFromGeos_XYZ()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaFromGeos_XYZ (
    const void * geos )
```

Converts a GEOS Geometry into a Geometry object [XYZ dims].

Parameters

<i>geos</i>	handle to GEOS Geometry
-------------	-------------------------

Returns

the pointer to the newly created Geometry object

See also

[gaiaFromGeos_XYZ_r](#), [gaiaToGeos](#), [gaiaFromGeos_XY](#), [gaiaFromGeos_XYM](#), [gaiaFromGeos_XYZM](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaFromGeos_XYZ\(\)](#) not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.32 gaiaFromGeos_XYZ_r()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaFromGeos_XYZ_r (
    const void * p_cache,
    const void * geos )
```

Converts a GEOS Geometry into a Geometry object [XYZ dims].

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geos</i>	handle to GEOS Geometry

Returns

the pointer to the newly created Geometry object

See also

[gaiaFromGeos_XYZ](#), [gaiaToGeos](#), [gaiaFromGeos_XY](#), [gaiaFromGeos_XYM](#), [gaiaFromGeos_XYZM](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaFromGeos_XYZ_r\(\)](#)
reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.33 [gaiaFromGeos_XYZM\(\)](#)

```
GAIGEO_DECLARE gaiaGeomCollPtr gaiaFromGeos\_XYZM (  
    const void * geos )
```

Converts a GEOS Geometry into a Geometry object [XYZM dims].

Parameters

<i>geos</i>	handle to GEOS Geometry
-------------	-------------------------

Returns

the pointer to the newly created Geometry object

See also

[gaiaFromGeos_XYZM_r](#), [gaiaToGeos](#), [gaiaFromGeos_XY](#), [gaiaFromGeos_XYZ](#), [gaiaFromGeos_XYM](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaFromGeos_XYZM\(\)](#)
not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.34 gaiaFromGeos_XYZM_r()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaFromGeos_XYZM_r (
    const void * p_cache,
    const void * geos )
```

Converts a GEOS Geometry into a Geometry object [XYZM dims].

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geos</i>	handle to GEOS Geometry

Returns

the pointer to the newly created Geometry object

See also

[gaiaFromGeos_XYZM](#), [gaiaToGeos](#), [gaiaFromGeos_XY](#), [gaiaFromGeos_XYZ](#), [gaiaFromGeos_XYM](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaFromGeos_XYZM_r\(\)](#)
reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.35 gaiaFromTWKB()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaFromTWKB (
    const void * p_cache,
    const unsigned char * twkb,
    int twkb_size,
    int srid )
```

Converts a compressed TWKB Geometry into a native binary Geometry.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>twkb</i>	pointer to TWKB geometry
<i>twkb_size</i>	size (in bytes) of the TWKB geometry
<i>srid</i>	the SRID of the returned Geometry

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaToTWKB](#)

Note

you are responsible to destroy the native geometry returned by [gaiaFromTWKB\(\)](#).

Remarks

RTTOPO support required.

6.5.2.36 [gaiaGeodesicArea\(\)](#)

```
GAIA GEO_DECLARE int gaiaGeodesicArea (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    double a,
    double b,
    int use_ellipsoid,
    double * area )
```

Measures the total Area for a Geometry object (geodesic)

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to Geometry object
<i>a</i>	major axis of the reference spheroid.
<i>b</i>	minor axis of the reference spheroid.
<i>use_ellipsoid</i>	if TRUE will measure the Area on the Ellipsoid, otherwise on the Sphere
<i>area</i>	on completion this variable will contain the measured area

Returns

0 on failure: any other value on success

See also

[gaiaGeomCollLength](#), [gaiaMeasureArea](#), [gaiaGeomCollArea](#)

Remarks

RTTOPO support required.

6.5.2.37 gaiaGeoHash()

```
GAIA GEO_DECLARE char* gaiaGeoHash (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    int precision )
```

Utility function: GeoHash.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	the input geometry.
<i>precision</i>	the expected precision: if <= 0 will be automatically determined.

Returns

NULL on failure: a null-terminated text string on success

Note

you are responsible to free (before or after) any text string returned by [gaiaGeoHash\(\)](#)

Remarks

RTTOPO support required.

6.5.2.38 gaiaGeomCollArea()

```
GAIA GEO_DECLARE int gaiaGeomCollArea (
    gaiaGeomCollPtr geom,
    double * area )
```

Measures the total Area for a Geometry object.

Parameters

<i>geom</i>	pointer to Geometry object
<i>area</i>	on completion this variable will contain the measured area

Returns

0 on failure: any other value on success

See also

[gaiaGeoCollArea_r](#), [gaiaGeomCollLength](#), [gaiaMeasureArea](#), [gaiaGeodesicArea](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

Examples

[demo1.c](#).

6.5.2.39 [gaiaGeomCollArea_r\(\)](#)

```
GAIA GEO_DECLARE int gaiaGeomCollArea\_r (  
    const void * p_cache,  
    gaiaGeomCollPtr geom,  
    double * area )
```

Measures the total Area for a Geometry object.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to Geometry object
<i>area</i>	on completion this variable will contain the measured area

Returns

0 on failure: any other value on success

See also

[gaiaGeoCollArea](#), [gaiaGeomCollLength](#), [gaiaMeasureArea](#), [gaiaGeodesicArea](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.40 [gaiaGeomCollBuffer\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeomCollBuffer (  
    gaiaGeomCollPtr geom,  
    double radius,  
    int points )
```

Spatial operator: Buffer.

Parameters

<i>geom</i>	the input Geometry object
<i>radius</i>	the buffer's radius
<i>points</i>	number of points (aka vertices) to be used in order to approximate a circular arc.

Returns

the pointer to newly created Geometry object representing the Buffer of input Geometry: NULL on failure.

See also

[gaiaGeomCollBuffer_r](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaGeomCollBuffer\(\)](#)
not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.41 `gaiaGeomCollBuffer_r()`

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaGeomCollBuffer_r (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    double radius,
    int points )
```

Spatial operator: Buffer.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	the input Geometry object
<i>radius</i>	the buffer's radius
<i>points</i>	number of points (aka vertices) to be used in order to approximate a circular arc.

Returns

the pointer to newly created Geometry object representing the Buffer of input Geometry: NULL on failure.

See also

[gaiaGeomCollBuffer](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaGeomCollBuffer_r\(\)](#)
reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.42 `gaiaGeomCollCentroid()`

```
GAIA GEO_DECLARE int gaiaGeomCollCentroid (
    gaiaGeomCollPtr geom,
    double * x,
    double * y )
```

Spatial operator: Centroid.

Parameters

<i>geom</i>	pointer to Geometry object.
<i>x</i>	on completion this variable will contain the centroid X coordinate
<i>y</i>	on completion this variable will contain the centroid Y coordinate

Returns

0 on failure: any other value on success

See also

[gaiaGeomCollCentroid_r](#), [gaiaRingCentroid](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.43 [gaiaGeomCollCentroid_r\(\)](#)

```
GAIA GEO_DECLARE int gaiaGeomCollCentroid\_r (  
    const void * p_cache,  
    gaiaGeomCollPtr geom,  
    double * x,  
    double * y )
```

Spatial operator: Centroid.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to Geometry object.
<i>x</i>	on completion this variable will contain the centroid X coordinate
<i>y</i>	on completion this variable will contain the centroid Y coordinate

Returns

0 on failure: any other value on success

See also

[gaiaGeomCollCentroid](#), [gaiaRingCentroid](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.44 `gaiaGeomCollContains()`

```
GAIAGEO_DECLARE int gaiaGeomCollContains (
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial relationship evaluation: Contains.

Parameters

<i>geom1</i>	the first Geometry object to be evaluated
<i>geom2</i>	the second Geometry object to be evaluated

Returns

0 if false: any other value if true

See also

[gaiaGeomCollContains_r](#), [gaiaGeomCollPreparedContains](#), [gaiaGeomCollEquals](#), [gaiaGeomCollDisjoint](#), [gaiaGeomCollIntersects](#), [gaiaGeomCollOverlaps](#), [gaiaGeomCollCrosses](#), [gaiaGeomCollWithin](#), [gaiaGeomCollTouches](#), [gaiaGeomCollRelate](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.45 `gaiaGeomCollContains_r()`

```
GAIAGEO_DECLARE int gaiaGeomCollContains_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial relationship evaluation: Contains.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object to be evaluated
<i>geom2</i>	the second Geometry object to be evaluated

Returns

0 if false: any other value if true

See also

[gaiaGeomCollContains](#), [gaiaGeomCollPreparedContains](#), [gaiaGeomCollEquals](#), [gaiaGeomCollDisjoint](#), [gaiaGeomCollIntersects](#), [gaiaGeomCollOverlaps](#), [gaiaGeomCollCrosses](#), [gaiaGeomCollWithin](#), [gaiaGeomCollTouches](#), [gaiaGeomCollRelate](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.46 [gaiaGeomCollCoveredBy\(\)](#)

```
GAIAGEO_DECLARE int gaiaGeomCollCoveredBy (  
    gaiaGeomCollPtr geom1,  
    gaiaGeomCollPtr geom2 )
```

Topology check: test if a Geometry is covered by another one.

Parameters

<i>geom1</i>	pointer to first input Geometry object.
<i>geom2</i>	pointer to second input Geometry object.

Returns

0 if false; any other value if *geom2* is *spatially covered by* *geom1*.

See also

[gaiaGeomCollCoveredBy_r](#), [gaiaGeomCollPreparedCoveredBy](#), [gaiaGeomCollCovers](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS-ADVANCED support required.

6.5.2.47 `gaiaGeomCollCoveredBy_r()`

```
GAIA GEO_DECLARE int gaiaGeomCollCoveredBy_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Topology check: test if a Geometry is covered by another one.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	pointer to first input Geometry object.
<i>geom2</i>	pointer to second input Geometry object.

Returns

0 if false; any other value if *geom2* is *spatially covered by* *geom1*.

See also

[gaiaGeomCollCoveredBy](#), [gaiaGeomCollPreparedCoveredBy](#), [gaiaGeomCollCovers](#)

Note

reentrant and thread-safe.

Remarks

GEOS-ADVANCED support required.

6.5.2.48 `gaiaGeomCollCovers()`

```
GAIA GEO_DECLARE int gaiaGeomCollCovers (
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Topology check: test if a Geometry covers another one.

Parameters

<i>geom1</i>	pointer to first input Geometry object.
<i>geom2</i>	pointer to second input Geometry object.

Returns

0 if false; any other value if geom1 *spatially covers* geom2.

See also

[gaiaGeomCollCovers_r](#), [gaiaGeomCollPreparedCovers](#), [gaiaGeomCollCoveredBy](#)

Note

not reentrant and thead unsafe.

Remarks

GEOS-ADVANCED support required.

6.5.2.49 gaiaGeomCollCovers_r()

```
GAIA GEO_DECLARE int gaiaGeomCollCovers_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Topology check: test if a Geometry covers another one.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	pointer to first input Geometry object.
<i>geom2</i>	pointer to second input Geometry object.

Returns

0 if false; any other value if geom1 *spatially covers* geom2.

See also

[gaiaGeomCollCovers](#), [gaiaGeomCollPreparedCovers](#), [gaiaGeomCollCoveredBy](#)

Note

reentrant and thead-safe.

Remarks

GEOS-ADVANCED support required.

6.5.2.50 `gaiaGeomCollCrosses()`

```
GAIAGEO_DECLARE int gaiaGeomCollCrosses (
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial relationship evaluation: Crosses.

Parameters

<i>geom1</i>	the first Geometry object to be evaluated
<i>geom2</i>	the second Geometry object to be evaluated

Returns

0 if false: any other value if true

See also

[gaiaGeomCollCrosses_r](#), [gaiaGeomCollPreparedCrosses](#), [gaiaGeomCollEquals](#), [gaiaGeomCollDisjoint](#), [gaiaGeomCollIntersects](#), [gaiaGeomCollOverlaps](#), [gaiaGeomCollContains](#), [gaiaGeomCollWithin](#), [gaiaGeomCollTouches](#), [gaiaGeomCollRelate](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.51 `gaiaGeomCollCrosses_r()`

```
GAIAGEO_DECLARE int gaiaGeomCollCrosses_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial relationship evaluation: Crosses.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object to be evaluated
<i>geom2</i>	the second Geometry object to be evaluated

Returns

0 if false: any other value if true

See also

[gaiaGeomCollCrosses](#), [gaiaGeomCollPreparedCrosses](#), [gaiaGeomCollEquals](#), [gaiaGeomCollDisjoint](#), [gaiaGeomCollIntersects](#), [gaiaGeomCollOverlaps](#), [gaiaGeomCollContains](#), [gaiaGeomCollWithin](#), [gaiaGeomCollTouches](#), [gaiaGeomCollRelate](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.52 [gaiaGeomCollDisjoint\(\)](#)

```
GAIAGEO_DECLARE int gaiaGeomCollDisjoint (  
    gaiaGeomCollPtr geom1,  
    gaiaGeomCollPtr geom2 )
```

Spatial relationship evaluation: Disjoint.

Parameters

<i>geom1</i>	the first Geometry object to be evaluated
<i>geom2</i>	the second Geometry object to be evaluated

Returns

0 if false: any other value if true

See also

[gaiaGeomCollDisjoint_r](#), [gaiaGeomCollEquals](#), [gaiaGeomCollIntersects](#), [gaiaGeomCollOverlaps](#), [gaiaGeomCollCrosses](#), [gaiaGeomCollContains](#), [gaiaGeomCollWithin](#), [gaiaGeomCollTouches](#), [gaiaGeomCollRelate](#)

Note

Obsolete: not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.53 `gaiaGeomCollDisjoint_r()`

```
GAIAGEO_DECLARE int gaiaGeomCollDisjoint_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial relationship evaluation: Disjoint.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object to be evaluated
<i>geom2</i>	the second Geometry object to be evaluated

Returns

0 if false: any other value if true

See also

[gaiaGeomCollDisjoint_r](#), [gaiaGeomCollEquals](#), [gaiaGeomCollIntersects](#), [gaiaGeomCollOverlaps](#), [gaiaGeomCollCrosses](#), [gaiaGeomCollContains](#), [gaiaGeomCollWithin](#), [gaiaGeomCollTouches](#), [gaiaGeomCollRelate](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.54 `gaiaGeomCollDistance()`

```
GAIAGEO_DECLARE int gaiaGeomCollDistance (
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2,
    double * dist )
```

Calculates the maximum distance intercurring between two Geometry objects.

Parameters

<i>geom1</i>	the first Geometry object
<i>geom2</i>	the second Geometry object
<i>dist</i>	on completion this variable will contain the calculated distance

Returns

0 on failure: any other value on success.

See also

[gaiaGeomCollDistance_r](#), [gaiaGeomCollPreparedDistance](#), [gaia3DDistance](#), [gaiaMaxDistance](#), [gaia3DMaxDistance](#)

Note

this function always computes the 2D cartesian distance.
not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.55 [gaiaGeomCollDistance_r\(\)](#)

```
GAIA GEO_DECLARE int gaiaGeomCollDistance\_r (  
    const void * p_cache,  
    gaiaGeomCollPtr geom1,  
    gaiaGeomCollPtr geom2,  
    double * dist )
```

Calculates the maximum distance intercurring between two Geometry objects.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object
<i>geom2</i>	the second Geometry object
<i>dist</i>	on completion this variable will contain the calculated distance

Returns

0 on failure: any other value on success.

See also

[gaiaGeomCollDistance](#), [gaiaGeomCollPreparedDistance](#), [gaia3DDistance](#), [gaiaMaxDistance](#), [gaia3DMaxDistance](#),
[gaiaGeomCollPreparedDistanceWithin](#)

Note

this function always computes the 2D cartesian distance.
reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.56 `gaiaGeomCollEquals()`

```
GAIAGEO_DECLARE int gaiaGeomCollEquals (
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial relationship evaluation: Equals.

Parameters

<i>geom1</i>	the first Geometry object to be evaluated
<i>geom2</i>	the second Geometry object to be evaluated

Returns

0 if false: any other value if true

See also

[gaiaGeomCollEquals_r](#), [gaiaGeomCollDisjoint](#), [gaiaGeomCollIntersects](#), [gaiaGeomCollOverlaps](#), [gaiaGeomCollCrosses](#), [gaiaGeomCollContains](#), [gaiaGeomCollWithin](#), [gaiaGeomCollTouches](#), [gaiaGeomCollRelate](#)

Note

Obsolete: not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.57 `gaiaGeomCollEquals_r()`

```
GAIAGEO_DECLARE int gaiaGeomCollEquals_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial relationship evaluation: Equals.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object to be evaluated
<i>geom2</i>	the second Geometry object to be evaluated

Returns

0 if false: any other value if true

See also

[gaiaGeomCollEquals](#), [gaiaGeomCollDisjoint](#), [gaiaGeomCollIntersects](#), [gaiaGeomCollOverlaps](#), [gaiaGeomCollCrosses](#), [gaiaGeomCollContains](#), [gaiaGeomCollWithin](#), [gaiaGeomCollTouches](#), [gaiaGeomCollRelate](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.58 [gaiaGeomCollIntersects\(\)](#)

```
GAIAGEO_DECLARE int gaiaGeomCollIntersects (  
    gaiaGeomCollPtr geom1,  
    gaiaGeomCollPtr geom2 )
```

Spatial relationship evaluation: Inteseacts.

Parameters

<i>geom1</i>	the first Geometry object to be evaluated
<i>geom2</i>	the second Geometry object to be evaluated

Returns

0 if false: any other value if true

See also

[gaiaGeomCollIntersects_r](#), [gaiaGeomCollPreparedIntersects](#), [gaiaGeomCollEquals](#), [gaiaGeomCollDisjoint](#), [gaiaGeomCollOverlaps](#), [gaiaGeomCollCrosses](#), [gaiaGeomCollContains](#), [gaiaGeomCollWithin](#), [gaiaGeomCollTouches](#), [gaiaGeomCollRelate](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.59 gaiaGeomCollIntersects_r()

```
GAIAGEO_DECLARE int gaiaGeomCollIntersects_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial relationship evaluation: Intersects.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object to be evaluated
<i>geom2</i>	the second Geometry object to be evaluated

Returns

0 if false: any other value if true

See also

[gaiaGeomCollIntersects](#), [gaiaGeomCollPreparedIntersects](#), [gaiaGeomCollEquals](#), [gaiaGeomCollDisjoint](#),
[gaiaGeomCollOverlaps](#), [gaiaGeomCollCrosses](#), [gaiaGeomCollContains](#), [gaiaGeomCollWithin](#), [gaiaGeomCollTouches](#),
[gaiaGeomCollRelate](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.60 gaiaGeomCollLength()

```
GAIAGEO_DECLARE int gaiaGeomCollLength (
    gaiaGeomCollPtr geom,
    double * length )
```

Measures the total Length for a Geometry object.

Parameters

<i>geom</i>	pointer to Geometry object
<i>length</i>	on completion this variable will contain the measured length

Returns

0 on failure: any other value on success

See also

[gaiaGeomCollLength_r](#), [gaiaGeomCollArea](#), [gaiaMeasureLength](#), [gaiaGeomCollLengthOrPerimeter](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.61 [gaiaGeomCollLength_r\(\)](#)

```
GAIA GEO_DECLARE int gaiaGeomCollLength\_r (  
    const void * p_cache,  
    gaiaGeomCollPtr geom,  
    double * length )
```

Measures the total Length for a Geometry object.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to Geometry object
<i>length</i>	on completion this variable will contain the measured length

Returns

0 on failure: any other value on success

See also

[gaiaGeomCollLength](#), [gaiaGeomCollArea](#), [gaiaMeasureLength](#), [gaiaGeomCollLengthOrPerimeter](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

Examples

[demo1.c](#).

6.5.2.62 gaiaGeomCollLengthOrPerimeter()

```
GAIAGEO_DECLARE int gaiaGeomCollLengthOrPerimeter (
    gaiaGeomCollPtr geom,
    int perimeter,
    double * length )
```

Measures the total Length or Perimeter for a Geometry object.

Parameters

<i>geom</i>	pointer to Geometry object
<i>perimeter</i>	if TRUE only Polygons will be considered, ignoring any Linestring the opposite if FALSE (considering only Linestrings and ignoring any Polygon)
<i>length</i>	on completion this variable will contain the measured length or perimeter

Returns

0 on failure: any other value on success

See also

[gaiaGeomCollLengthOrPerimeter_r](#), [gaiaGeomCollArea](#), [gaiaMeasureLength](#), [gaiaGeomCollLength](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.63 `gaiaGeomCollLengthOrPerimeter_r()`

```
GAIA GEO_DECLARE int gaiaGeomCollLengthOrPerimeter_r (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    int perimeter,
    double * length )
```

Measures the total Length or Perimeter for a Geometry object.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to Geometry object
<i>perimeter</i>	if TRUE only Polygons will be considered, ignoring any Linestring the opposite if FALSE (considering only Linestrings and ignoring any Polygon)
<i>length</i>	on completion this variable will contain the measured length or perimeter

Returns

0 on failure: any other value on success

See also

[gaiaGeomCollLengthOrPerimeter](#), [gaiaGeomCollArea](#), [gaiaMeasureLength](#), [gaiaGeomCollLength](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.64 `gaiaGeomCollOverlaps()`

```
GAIA GEO_DECLARE int gaiaGeomCollOverlaps (
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial relationship evaluation: Overlaps.

Parameters

<i>geom1</i>	the first Geometry object to be evaluated
<i>geom2</i>	the second Geometry object to be evaluated

Returns

0 if false: any other value if true

See also

[gaiaGeomCollOverlaps_r](#), [gaiaGeomCollPreparedOverlaps](#), [gaiaGeomCollEquals](#), [gaiaGeomCollDisjoint](#), [gaiaGeomCollIntersects](#), [gaiaGeomCollCrosses](#), [gaiaGeomCollContains](#), [gaiaGeomCollWithin](#), [gaiaGeomCollTouches](#), [gaiaGeomCollRelate](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.65 gaiaGeomCollOverlaps_r()

```
GAIAGEO_DECLARE int gaiaGeomCollOverlaps_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial relationship evaluation: Overlaps.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object to be evaluated
<i>geom2</i>	the second Geometry object to be evaluated

Returns

0 if false: any other value if true

See also

[gaiaGeomCollOverlaps](#), [gaiaGeomCollPreparedOverlaps](#), [gaiaGeomCollEquals](#), [gaiaGeomCollDisjoint](#), [gaiaGeomCollIntersects](#), [gaiaGeomCollCrosses](#), [gaiaGeomCollContains](#), [gaiaGeomCollWithin](#), [gaiaGeomCollTouches](#), [gaiaGeomCollRelate](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.66 gaiaGeomCollPreparedContains()

```

GAIAGEO_DECLARE int gaiaGeomCollPreparedContains (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    unsigned char * blob1,
    int size1,
    gaiaGeomCollPtr geom2,
    unsigned char * blob2,
    int size2 )

```

Spatial relationship evaluation: Contains (GEOSPreparedGeometry)

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object to be evaluated
<i>blob1</i>	the BLOB corresponding to the first Geometry
<i>size1</i>	the size (in bytes) of the first BLOB
<i>geom2</i>	the second Geometry object to be evaluated
<i>blob2</i>	the BLOB corresponding to the second Geometry
<i>size2</i>	the size (in bytes) of the second BLOB

Returns

0 if false: any other value if true

See also

[gaiaGeomCollContains](#), [gaiaGeomCollContains_r](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.67 gaiaGeomCollPreparedCoveredBy()

```

GAIAGEO_DECLARE int gaiaGeomCollPreparedCoveredBy (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    unsigned char * blob1,
    int size1,
    gaiaGeomCollPtr geom2,
    unsigned char * blob2,
    int size2 )

```

Topology check: test if a Geometry is covered by another one (GEOSPreparedGeometry)

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	pointer to first input Geometry object.
<i>blob1</i>	the BLOB corresponding to the first Geometry
<i>size1</i>	the size (in bytes) of the first BLOB
<i>geom2</i>	pointer to second input Geometry object.
<i>blob2</i>	the BLOB corresponding to the second Geometry
<i>size2</i>	the size (in bytes) of the second BLOB

Returns

0 if false; any other value if *geom2* is *spatially covered by* *geom1*.

See also

[gaiaGeomCollCoveredBy](#), [gaiaGeomCollCoveredBy_r](#), [gaiaGeomCollCovers](#)

Note

reentrant and thread-safe.

Remarks

GEOS-ADVANCED support required.

6.5.2.68 [gaiaGeomCollPreparedCovers\(\)](#)

```
GAIA GEO_DECLARE int gaiaGeomCollPreparedCovers (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    unsigned char * blob1,
    int size1,
    gaiaGeomCollPtr geom2,
    unsigned char * blob2,
    int size2 )
```

Topology check: test if a Geometry covers another one (GEOSPreparedGeometry)

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	pointer to first input Geometry object.
<i>blob1</i>	the BLOB corresponding to the first Geometry
<i>size1</i>	the size (in bytes) of the first BLOB
<i>geom2</i>	pointer to second input Geometry object.
<i>blob2</i>	the BLOB corresponding to the second Geometry
<i>size2</i>	the size (in bytes) of the second BLOB

Returns

0 if false; any other value if geom1 *spatially covers* geom2.

Note

reentrant and thread-safe.

See also

[gaiaGeomCollCovers](#), [gaiaGeomCollCovers_r](#)

Remarks

GEOS-ADVANCED support required.

6.5.2.69 [gaiaGeomCollPreparedCrosses\(\)](#)

```
GAIA GEO_DECLARE int gaiaGeomCollPreparedCrosses (  
    const void * p_cache,  
    gaiaGeomCollPtr geom1,  
    unsigned char * blob1,  
    int size1,  
    gaiaGeomCollPtr geom2,  
    unsigned char * blob2,  
    int size2 )
```

Spatial relationship evaluation: Crosses (GEOSPreparedGeometry)

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object to be evaluated
<i>blob1</i>	the BLOB corresponding to the first Geometry
<i>size1</i>	the size (in bytes) of the first BLOB
<i>geom2</i>	the second Geometry object to be evaluated
<i>blob2</i>	the BLOB corresponding to the second Geometry
<i>size2</i>	the size (in bytes) of the second BLOB

Returns

0 if false; any other value if true

Note

reentrant and thread-safe.

See also

[gaiaGeomCollCrosses](#), [gaiaGeomCollCrosses_r](#)

Remarks

GEOS support required.

6.5.2.70 `gaiaGeomCollPreparedDisjoint()`

```
GAIA GEO_DECLARE int gaiaGeomCollPreparedDisjoint (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    unsigned char * blob1,
    int size1,
    gaiaGeomCollPtr geom2,
    unsigned char * blob2,
    int size2 )
```

Spatial relationship evaluation: Disjoint (GEOSPreparedGeometry)

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object to be evaluated
<i>blob1</i>	the BLOB corresponding to the first Geometry
<i>size1</i>	the size (in bytes) of the first BLOB
<i>geom2</i>	the second Geometry object to be evaluated
<i>blob2</i>	the BLOB corresponding to the second Geometry
<i>size2</i>	the size (in bytes) of the second BLOB

Returns

0 if false: any other value if true

See also

[gaiaGeomCollDisjoint](#), [gaiaGeomCollDisjoint_r](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.71 `gaiaGeomCollPreparedDistance()`

```

GAIA GEO_DECLARE int gaiaGeomCollPreparedDistance (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    unsigned char * blob1,
    int size1,
    gaiaGeomCollPtr geom2,
    unsigned char * blob2,
    int size2,
    double * dist )

```

Spatial relationship evaluation: Distance (GEOSPreparedGeometry)

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object to be evaluated
<i>blob1</i>	the BLOB corresponding to the first Geometry
<i>size1</i>	the size (in bytes) of the first BLOB
<i>geom2</i>	the second Geometry object to be evaluated
<i>blob2</i>	the BLOB corresponding to the second Geometry
<i>size2</i>	the size (in bytes) of the second BLOB
<i>dist</i>	on completion this variable will contain the calculated distance

Returns

0 if false: any other value if true

See also

[gaiaGeomCollDistance](#), [gaiaGeomCollDistance_r](#), [gaiaGeomCollPreparedDistanceWithin](#)

Note

reentrant and thread-safe.

Remarks

GEOS_3100 support required.

6.5.2.72 `gaiaGeomCollPreparedDistanceWithin()`

```

GAIA GEO_DECLARE int gaiaGeomCollPreparedDistanceWithin (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    unsigned char * blob1,
    int size1,
    gaiaGeomCollPtr geom2,
    unsigned char * blob2,
    int size2,
    double dist )

```

Spatial operator: DistanceWithin (GEOSPreparedGeometry)

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object to be evaluated
<i>blob1</i>	the BLOB corresponding to the first Geometry
<i>size1</i>	the size (in bytes) of the first BLOB
<i>geom2</i>	the second Geometry object to be evaluated
<i>blob2</i>	the BLOB corresponding to the second Geometry
<i>size2</i>	the size (in bytes) of the second BLOB
<i>dist</i>	the max distance to be checked

Returns

0 if false: any other value if true

See also

[gaiaGeomCollDistance](#), [gaiaGeomCollDistance_r](#), [gaiaGeomCollPreparedDistance](#)

Note

reentrant and thread-safe.

Remarks

GEOS_3100 support required.

6.5.2.73 [gaiaGeomCollPreparedIntersects\(\)](#)

```

GAIAGEO_DECLARE int gaiaGeomCollPreparedIntersects (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    unsigned char * blob1,
    int size1,
    gaiaGeomCollPtr geom2,
    unsigned char * blob2,
    int size2 )

```

Spatial relationship evaluation: Intersects (GEOSPreparedGeometry)

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object to be evaluated
<i>blob1</i>	the BLOB corresponding to the first Geometry
<i>size1</i>	the size (in bytes) of the first BLOB
<i>geom2</i>	the second Geometry object to be evaluated
<i>blob2</i>	the BLOB corresponding to the second Geometry
<i>size2</i>	the size (in bytes) of the second BLOB

Returns

0 if false: any other value if true

See also

[gaiaGeomCollIntersects](#), [gaiaGeomCollIntersects_r](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.74 gaiaGeomCollPreparedOverlaps()

```
GAIA GEO_DECLARE int gaiaGeomCollPreparedOverlaps (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    unsigned char * blob1,
    int size1,
    gaiaGeomCollPtr geom2,
    unsigned char * blob2,
    int size2 )
```

Spatial relationship evaluation: Overlaps (GEOSPreparedGeometry)

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object to be evaluated
<i>blob1</i>	the BLOB corresponding to the first Geometry
<i>size1</i>	the size (in bytes) of the first BLOB
<i>geom2</i>	the second Geometry object to be evaluated
<i>blob2</i>	the BLOB corresponding to the second Geometry
<i>size2</i>	the size (in bytes) of the second BLOB

Returns

0 if false: any other value if true

See also

[gaiaGeomCollOverlaps](#), [gaiaGeomCollOverlaps_r](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.75 gaiaGeomCollPreparedTouches()

```
GAIA GEO_DECLARE int gaiaGeomCollPreparedTouches (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    unsigned char * blob1,
    int size1,
    gaiaGeomCollPtr geom2,
    unsigned char * blob2,
    int size2 )
```

Spatial relationship evaluation: Touches (GEOSPreparedGeometry)

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object to be evaluated
<i>blob1</i>	the BLOB corresponding to the first Geometry
<i>size1</i>	the size (in bytes) of the first BLOB
<i>geom2</i>	the second Geometry object to be evaluated
<i>blob2</i>	the BLOB corresponding to the second Geometry
<i>size2</i>	the size (in bytes) of the second BLOB

Returns

0 if false: any other value if true

See also

[gaiaGeomCollTouches](#), [gaiaGeomCollTouches_r](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.76 gaiaGeomCollPreparedWithin()

```

GAIAGEO_DECLARE int gaiaGeomCollPreparedWithin (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    unsigned char * blob1,
    int size1,
    gaiaGeomCollPtr geom2,
    unsigned char * blob2,
    int size2 )

```

Spatial relationship evaluation: Within (GEOSPreparedGeometry)

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object to be evaluated
<i>blob1</i>	the BLOB corresponding to the first Geometry
<i>size1</i>	the size (in bytes) of the first BLOB
<i>geom2</i>	the second Geometry object to be evaluated
<i>blob2</i>	the BLOB corresponding to the second Geometry
<i>size2</i>	the size (in bytes) of the second BLOB

Returns

0 if false: any other value if true

See also

[gaiaGeomCollWithin](#), [gaiaGeomCollWithin_r](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.77 gaiaGeomCollRelate()

```

GAIAGEO_DECLARE int gaiaGeomCollRelate (
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2,
    const char * pattern )

```

Spatial relationship evaluation: Relate.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object to be evaluated
<i>geom2</i>	the second Geometry object to be evaluated
<i>pattern</i>	intersection matrix pattern [DE-9IM]

Returns

0 if false: any other value if true

See also

[gaiaGeomCollRelate_r](#), [gaiaGeomCollRelateBoundaryNodeRule](#), [gaiaGeomCollEquals](#), [gaiaGeomCollDisjoint](#), [gaiaGeomCollIntersects](#), [gaiaGeomCollOverlaps](#), [gaiaGeomCollCrosses](#), [gaiaGeomCollContains](#), [gaiaGeomCollWithin](#), [gaiaIntersectionMatrixPatternMatch](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.78 [gaiaGeomCollRelate_r\(\)](#)

```
GAIAGEO_DECLARE int gaiaGeomCollRelate_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2,
    const char * pattern )
```

Spatial relationship evaluation: Relate.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object to be evaluated
<i>geom2</i>	the second Geometry object to be evaluated
<i>pattern</i>	intersection matrix pattern [DE-9IM]

Returns

0 if false: any other value if true

See also

[gaiaGeomCollRelate](#), [gaiaGeomCollRelateBoundaryNodeRule_r](#), [gaiaGeomCollEquals](#), [gaiaGeomCollDisjoint](#), [gaiaGeomCollIntersects](#), [gaiaGeomCollOverlaps](#), [gaiaGeomCollCrosses](#), [gaiaGeomCollContains](#), [gaiaGeomCollWithin](#), [gaiaIntersectionMatrixPatternMatch](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.79 [gaiaGeomCollRelateBoundaryNodeRule\(\)](#)

```
GAIAGEO_DECLARE char* gaiaGeomCollRelateBoundaryNodeRule (
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2,
    int mode )
```

Spatial relationship evaluation: Relate Boundary Node Rule.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object to be evaluated
<i>geom2</i>	the second Geometry object to be evaluated
<i>mode</i>	can be one of: 1=OGC/MOD2 (default); 2=Endpoint; 3=MultivalentEndpoint; 4=MonovalentEndpoint

Returns

a DE-9IM intersection matrix; or NULL on invalid geometries.

See also

[gaiaGeomCollRelate](#), [gaiaGeomCollRelateBoundaryNodeRule_r](#), [gaiaGeomCollEquals](#), [gaiaGeomCollDisjoint](#), [gaiaGeomCollIntersects](#), [gaiaGeomCollOverlaps](#), [gaiaGeomCollCrosses](#), [gaiaGeomCollContains](#), [gaiaGeomCollWithin](#), [gaiaIntersectionMatrixPatternMatch](#)

Note

you are responsible to destroy (before or after) the intersection matrix returned by [gaiaGeomCollRelateBoundaryNodeRule\(\)](#)
not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.80 `gaiaGeomCollRelateBoundaryNodeRule_r()`

```
GAIAGEO_DECLARE char* gaiaGeomCollRelateBoundaryNodeRule_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2,
    int mode )
```

Spatial relationship evaluation: Relate Boundary Node Rule.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object to be evaluated
<i>geom2</i>	the second Geometry object to be evaluated
<i>mode</i>	can be one of: 1=OGC/MOD2 (default); 2=Endpoint; 3=MultivalentEndpoint; 4=MonovalentEndpoint

Returns

a DE-9IM intersection matrix; or NULL on invalid geometries.

See also

[gaiaGeomCollRelate_r](#), [gaiaGeomCollRelateBoundaryNodeRule](#), [gaiaGeomCollEquals](#), [gaiaGeomCollDisjoint](#), [gaiaGeomCollIntersects](#), [gaiaGeomCollOverlaps](#), [gaiaGeomCollCrosses](#), [gaiaGeomCollContains](#), [gaiaGeomCollWithin](#), [gaiaIntersectionMatrixPatternMatch](#)

Note

you are responsible to destroy (before or after) the intersection matrix returned by `gaiaGeomCollRelateBoundaryNodeRule_r()`
reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.81 `gaiaGeomCollSimplify()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeomCollSimplify (
    gaiaGeomCollPtr geom,
    double tolerance )
```

Spatial operator: Simplify.

Parameters

<i>geom</i>	the input Geometry object
<i>tolerance</i>	approximation threshold

Returns

the pointer to newly created Geometry object representing the simplified Geometry [applying the Douglas-Peucker algorithm]: NULL on failure.

See also

[gaiaGeomCollSimplify_r](#), [gaiaFreeGeomColl](#), [gaiaGeomCollSimplifyPreserveTopology](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaGeomCollSimplify\(\)](#) not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.82 [gaiaGeomCollSimplify_r\(\)](#)

```
GAIA_GEO_DECLARE gaiaGeomCollPtr gaiaGeomCollSimplify\_r (  
    const void * p_cache,  
    gaiaGeomCollPtr geom,  
    double tolerance )
```

Spatial operator: Simplify.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	the input Geometry object
<i>tolerance</i>	approximation threshold

Returns

the pointer to newly created Geometry object representing the simplified Geometry [applying the Douglas-Peucker algorithm]: NULL on failure.

See also

[gaiaGeomCollSimplify](#), [gaiaFreeGeomColl](#), [gaiaGeomCollSimplifyPreserveTopology](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaGeomCollSimplify_r\(\)](#) reentrant and thread safe.

Remarks

GEOS support required.

6.5.2.83 `gaiaGeomCollSimplifyPreserveTopology()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeomCollSimplifyPreserveTopology (
    gaiaGeomCollPtr geom,
    double tolerance )
```

Spatial operator: Simplify [preserving topology].

Parameters

<i>geom</i>	the input Geometry object
<i>tolerance</i>	approximation threshold

Returns

the pointer to newly created Geometry object representing the simplified Geometry [applying the Douglas-Peucker algorithm]: NULL on failure.

See also

[gaiaGeomCollSimplifyPreserveTopology_r](#), [gaiaFreeGeomColl](#), [gaiaGeomCollSimplify](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaGeomCollSimplify\(\)](#)
not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.84 `gaiaGeomCollSimplifyPreserveTopology_r()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeomCollSimplifyPreserveTopology_r (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    double tolerance )
```

Spatial operator: Simplify [preserving topology].

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	the input Geometry object
<i>tolerance</i>	approximation threshold

Returns

the pointer to newly created Geometry object representing the simplified Geometry [applying the Douglas-Peucker algorithm]: NULL on failure.

See also

[gaiaGeomCollSimplifyPreserveTopology](#), [gaiaFreeGeomColl](#), [gaiaGeomCollSimplify](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaGeomCollSimplify_r\(\)](#)
reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.85 [gaiaGeomCollTouches\(\)](#)

```
GAIAGEO_DECLARE int gaiaGeomCollTouches (
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial relationship evaluation: Touches.

Parameters

<i>geom1</i>	the first Geometry object to be evaluated
<i>geom2</i>	the second Geometry object to be evaluated

Returns

0 if false: any other value if true

See also

[gaiaGeomCollTouches_r](#), [gaiaGeomCollPreparedTouches](#), [gaiaGeomCollEquals](#), [gaiaGeomCollDisjoint](#),
[gaiaGeomCollIntersects](#), [gaiaGeomCollOverlaps](#), [gaiaGeomCollCrosses](#), [gaiaGeomCollContains](#), [gaiaGeomCollWithin](#),
[gaiaGeomCollRelate](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.86 gaiaGeomCollTouches_r()

```
GAIAGEO_DECLARE int gaiaGeomCollTouches_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial relationship evaluation: Touches.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object to be evaluated
<i>geom2</i>	the second Geometry object to be evaluated

Returns

0 if false: any other value if true

See also

[gaiaGeomCollTouches](#), [gaiaGeomCollPreparedTouches](#), [gaiaGeomCollEquals](#), [gaiaGeomCollDisjoint](#), [gaiaGeomCollIntersects](#), [gaiaGeomCollOverlaps](#), [gaiaGeomCollCrosses](#), [gaiaGeomCollContains](#), [gaiaGeomCollWithin](#), [gaiaGeomCollRelate](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.87 gaiaGeomCollWithin()

```
GAIAGEO_DECLARE int gaiaGeomCollWithin (
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial relationship evaluation: Within.

Parameters

<i>geom1</i>	the first Geometry object to be evaluated
<i>geom2</i>	the second Geometry object to be evaluated

Returns

0 if false: any other value if true

See also

[gaiaGeomCollWithin_r](#), [gaiaGeomCollPreparedWithin](#), [gaiaGeomCollEquals](#), [gaiaGeomCollDisjoint](#), [gaiaGeomCollIntersects](#), [gaiaGeomCollOverlaps](#), [gaiaGeomCollCrosses](#), [gaiaGeomCollContains](#), [gaiaGeomCollTouches](#), [gaiaGeomCollRelate](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.88 [gaiaGeomCollWithin_r\(\)](#)

```
GAIA_GEO_DECLARE int gaiaGeomCollWithin_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial relationship evaluation: Within.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object to be evaluated
<i>geom2</i>	the second Geometry object to be evaluated

Returns

0 if false: any other value if true

See also

[gaiaGeomCollWithin](#), [gaiaGeomCollPreparedWithin](#), [gaiaGeomCollEquals](#), [gaiaGeomCollDisjoint](#), [gaiaGeomCollIntersects](#), [gaiaGeomCollOverlaps](#), [gaiaGeomCollCrosses](#), [gaiaGeomCollContains](#), [gaiaGeomCollTouches](#), [gaiaGeomCollRelate](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.89 gaiaGeometryDifference()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeometryDifference (
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial operator: Difference.

Parameters

<i>geom1</i>	the first Geometry object
<i>geom2</i>	the second Geometry object

Returns

the pointer to newly created Geometry object representing the geometry Difference of both input Geometries:
NULL on failure.

See also

[gaiaGeometryDifference_r](#), [gaiaGeometrySymDifference](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaGeometryDifference\(\)](#)
not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.90 gaiaGeometryDifference_r()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeometryDifference_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial operator: Difference.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object
<i>geom2</i>	the second Geometry object

Returns

the pointer to newly created Geometry object representing the geometry Difference of both input Geometries: NULL on failure.

See also

[gaiaGeometryDifference](#), [gaiaGeometrySymDifference](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaGeometryDifference_r\(\)](#) reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.91 [gaiaGeometryIntersection\(\)](#)

```
GAIA_GEO_DECLARE gaiaGeomCollPtr gaiaGeometryIntersection (
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial operator: Intersection.

Parameters

<i>geom1</i>	the first Geometry object
<i>geom2</i>	the second Geometry object

Returns

the pointer to newly created Geometry object representing the geometry Intersection of both input Geometries: NULL on failure.

See also

[gaiaGeometryIntersection_r](#), [gaiaFreeGeomColl](#), [gaiaGeometryUnion](#), [gaiaGeometryDifference](#), [gaiaGeometrySymDifference](#), [gaiaBoundary](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaGeometryIntersection\(\)](#)
not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.92 gaiaGeometryIntersection_r()

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaGeometryIntersection_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial operator: Intersection.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object
<i>geom2</i>	the second Geometry object

Returns

the pointer to newly created Geometry object representing the geometry Intersection of both input Geometries: NULL on failure.

See also

[gaiaGeometryIntersection](#), [gaiaFreeGeomColl](#), [gaiaGeometryUnion](#), [gaiaGeometryDifference](#), [gaiaGeometrySymDifference](#), [gaiaBoundary](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaGeometryIntersection_r\(\)](#)
reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.93 gaiaGeometrySymDifference()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeometrySymDifference (
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial operator: SymDifference.

Parameters

<i>geom1</i>	the first Geometry object
<i>geom2</i>	the second Geometry object

Returns

the pointer to newly created Geometry object representing the geometry SymDifference of both input Geometries: NULL on failure.

See also

[gaiaGeometrySymDifference_r](#), [gaiaGeometryDifference](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaGeometrySymDifference\(\)](#)
not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.94 [gaiaGeometrySymDifference_r\(\)](#)

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaGeometrySymDifference\_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial operator: SymDifference.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object
<i>geom2</i>	the second Geometry object

Returns

the pointer to newly created Geometry object representing the geometry SymDifference of both input Geometries: NULL on failure.

See also

[gaiaGeometrySymDifference](#), [gaiaGeometryDifference](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaGeometrySymDifference_r\(\)](#) reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.95 gaiaGeometryUnion()

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaGeometryUnion (
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial operator: Union.

Parameters

<i>geom1</i>	the first Geometry object
<i>geom2</i>	the second Geometry object

Returns

the pointer to newly created Geometry object representing the geometry Union of both input Geometries: NULL on failure.

See also

[gaiaGeometryUnion_r](#), [gaiaFreeGeomColl](#), [gaiaUnaryUnion](#), [gaiaUnionCascaded](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaGeometryUnion\(\)](#) not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.96 gaiaGeometryUnion_r()

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaGeometryUnion_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial operator: Union.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object
<i>geom2</i>	the second Geometry object

Returns

the pointer to newly created Geometry object representing the geometry Union of both input Geometries: NULL on failure.

See also

[gaiaGeometryUnion](#), [gaiaFreeGeomColl](#), [gaiaUnaryUnion](#), [gaiaUnionCascaded](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaGeometryUnion_r\(\)](#) reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.97 [gaiaGeosConcaveHull\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeosConcaveHull (
    gaiaGeomCollPtr geom,
    double ratio,
    int allow_holes )
```

Concave Hull (the GEOS way)

Parameters

<i>geom</i>	pointer to input Geometry object.
<i>ratio</i>	an index ranging from 1.0 (Convex Hull) to 0.0 (maximum concaveness).
<i>allow_holes</i>	if FALSE any interior hole will be suppressed.

Returns

the pointer to newly created Geometry object (always of the Polygon type): NULL on failure. NULL will be returned if any argument is invalid.

See also

[gaiaGeosConcaveHull_r](#), [gaiaFreeGeomColl](#), [gaiaConcaveHull](#), [gaiaConcaveHull_r](#)

Note

This function is directly based on GEOSConcaveHull().

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaGeosConcaveHull\(\)](#)
not reentrant and thread unsafe.

Remarks

GEOS_3110 support required.

6.5.2.98 [gaiaGeosConcaveHull_r\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeosConcaveHull\_r (  
    const void * p_cache,  
    gaiaGeomCollPtr geom,  
    double ratio,  
    int allow_holes )
```

Concave Hull (the GEOS way)

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to input Geometry object.
<i>ratio</i>	an index ranging from 1.0 (Convex Hull) to 0.0 (maximum concaveness).
<i>allow_holes</i>	if FALSE any interior hole will be suppressed.

Returns

the pointer to newly created Geometry object (always of the Polygon type): NULL on failure.
NULL will be returned if any argument is invalid.

See also

[gaiaGeosConcaveHull](#), [gaiaFreeGeomColl](#), [gaiaConcaveHull](#), [gaiaConcaveHull_r](#)

Note

This function is directly based on GEOSConcaveHull_r().

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaGeosConcaveHull_r\(\)](#)
reentrant and thread-safe.

Remarks

GEOS_3110 support required.

6.5.2.99 `gaiaGeosDensify()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeosDensify (
    gaiaGeomCollPtr geom,
    double tolerance )
```

Densifies a geometry using a given distance tolerance.

Parameters

<i>geom</i>	pointer to input Geometry object.
<i>tolerance</i>	the distance tolerance to densify.

Returns

the pointer to newly created Geometry object: NULL on failure.
NULL will be returned if any argument is invalid.

See also

[gaiaGeosDensify_r](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaGeosDensify\(\)](#)
not reentrant and thread unsafe.

Remarks

GEOS_3100 support required.

6.5.2.100 `gaiaGeosDensify_r()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeosDensify_r (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    double tolerance )
```

Densifies a geometry using a given distance tolerance.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to input Geometry object.
<i>tolerance</i>	the distance tolerance to densify.

Returns

the pointer to newly created Geometry object: NULL on failure.
NULL will be returned if any argument is invalid.

See also

[gaiaGeosDensify](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaGeosDensify_r\(\)](#)
reentrant and thread-safe.

Remarks

GEOS_3100 support required.

6.5.2.101 [gaiaGeosMakeValid\(\)](#)

```
GAIA_GEO_DECLARE gaiaGeomCollPtr gaiaGeosMakeValid (  
    gaiaGeomCollPtr geom,  
    int keep_discarded )
```

Utility function: [GeosMakeValid](#).

Parameters

<i>geom</i>	the input Geometry object.
<i>keep_discarded</i>	boolean: if FALSE all geometry components that collapse to a lower dimensionality, for example a one-point linestring, will be dropped.

Returns

the pointer to newly created Geometry object: NULL on failure.
this function will attempt to create a valid representation of a given invalid geometry using the GEOS own "method=structure".

See also

[gaiaFreeGeomColl](#), [gaiaMakeValid](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaGeosMakeValid\(\)](#)

Remarks

GEOS_3100 support required.

6.5.2.102 [gaiaGeosMakeValid_r\(\)](#)

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaGeosMakeValid\_r (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    int keep_discarded )
```

Utility function: [GeosMakeValid](#).

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	the input Geometry object.
<i>keep_discarded</i>	boolean: if FALSE all geometry components that collapse to a lower dimensionality, for example a one-point linestring, will be dropped.

Returns

the pointer to newly created Geometry object: NULL on failure.
this function will attempt to create a valid representation of a given invalid geometry using the GEOS own "method=structure".

See also

[gaiaFreeGeomColl](#), [gaiaMakeValid](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaGeosMakeValid\(\)](#)

Remarks

GEOS_3100 support required.

6.5.2.103 `gaiaGetGeosAuxErrorMsg()`

```
GAIA GEO_DECLARE const char* gaiaGetGeosAuxErrorMsg (
    void )
```

Return the latest GEOS (auxiliary) error message (if any)

Returns

the latest GEOS (auxiliary) error message: an empty string if no error was previously found.

See also

[gaiaGetGeosAuxErrorMsg_r](#), [gaiaResetGeosMsg](#), [gaiaGetGeosErrorMsg](#), [gaiaGetGeosWarningMsg](#),
[gaiaSetGeosErrorMsg](#), [gaiaSetGeosWarningMsg](#), [gaiaSetGeosAuxErrorMsg](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.104 `gaiaGetGeosAuxErrorMsg_r()`

```
GAIA GEO_DECLARE const char* gaiaGetGeosAuxErrorMsg_r (
    const void * p_cache )
```

Return the latest GEOS (auxiliary) error message (if any)

Parameters

<code>p_cache</code>	a memory pointer returned by spatialite_alloc_connection()
----------------------	--

Returns

the latest GEOS (auxiliary) error message: an empty string if no error was previously found.

See also

[gaiaGetGeosAuxErrorMsg](#), [gaiaResetGeosMsg](#), [gaiaGetGeosErrorMsg](#), [gaiaGetGeosWarningMsg](#),
[gaiaSetGeosErrorMsg](#), [gaiaSetGeosWarningMsg](#), [gaiaSetGeosAuxErrorMsg](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.105 gaiaGetGeosErrorMsg()

```
GAIAGEO_DECLARE const char* gaiaGetGeosErrorMsg (  
    void )
```

Return the latest GEOS error message (if any)

Returns

the latest GEOS error message: an empty string if no error was previously found.

See also

[gaiaGetGeosErrorMsg_r](#), [gaiaResetGeosMsg](#), [gaiaGetGeosWarningMsg](#), [gaiaGetGeosAuxErrorMsg](#),
[gaiaSetGeosErrorMsg](#), [gaiaSetGeosWarningMsg](#), [gaiaSetGeosAuxErrorMsg](#), [gaiaCriticalPointFromGEOSmsg](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.106 gaiaGetGeosErrorMsg_r()

```
GAIAGEO_DECLARE const char* gaiaGetGeosErrorMsg_r (  
    const void * p_cache )
```

Return the latest GEOS error message (if any)

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
----------------	--

Returns

the latest GEOS error message: an empty string if no error was previously found.

See also

[gaiaGetGeosErrorMsg](#), [gaiaResetGeosMsg](#), [gaiaGetGeosWarningMsg](#), [gaiaGetGeosAuxErrorMsg](#), [gaiaSetGeosErrorMsg](#), [gaiaSetGeosWarningMsg](#), [gaiaSetGeosAuxErrorMsg](#), [gaiaCriticalPointFromGEOSmsg](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.107 [gaiaGetGeosWarningMsg\(\)](#)

```
GAIAGEO_DECLARE const char* gaiaGetGeosWarningMsg (  
    void )
```

Return the latest GEOS warning message (if any)

Returns

the latest GEOS warning message: an empty string if no warning was previously found.

See also

[gaiaGetGeosWarningMsg_r](#), [gaiaResetGeosMsg](#), [gaiaGetGeosErrorMsg](#), [gaiaGetGeosAuxErrorMsg](#), [gaiaSetGeosErrorMsg](#), [gaiaSetGeosWarningMsg](#), [gaiaSetGeosAuxErrorMsg](#), [gaiaCriticalPointFromGEOSmsg](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.108 [gaiaGetGeosWarningMsg_r\(\)](#)

```
GAIAGEO_DECLARE const char* gaiaGetGeosWarningMsg_r (  
    const void * p_cache )
```

Return the latest GEOS warning message (if any)

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
----------------	--

Returns

the latest GEOS warning message: an empty string if no warning was previously found.

See also

[gaiaGetGeosWarningMsg](#), [gaiaResetGeosMsg](#), [gaiaGetGeosErrorMsg](#), [gaiaGetGeosAuxErrorMsg](#),
[gaiaSetGeosErrorMsg](#), [gaiaSetGeosWarningMsg](#), [gaiaSetGeosAuxErrorMsg](#), [gaiaCriticalPointFromGEOSmsg](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.109 **gaiaGetPointOnSurface()**

```
GAIA GEO_DECLARE int gaiaGetPointOnSurface (
    gaiaGeomCollPtr geom,
    double * x,
    double * y )
```

Spatial operator: PointOnSurface.

Parameters

<i>geom</i>	pointer to Geometry object.
<i>x</i>	on completion this variable will contain the Point X coordinate
<i>y</i>	on completion this variable will contain the Point Y coordinate

Returns

0 on failure: any other value on success

See also

[gaiaGetPointOnSurface_r](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.110 gaiaGetPointOnSurface_r()

```
GAIAGEO_DECLARE int gaiaGetPointOnSurface_r (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    double * x,
    double * y )
```

Spatial operator: PointOnSurface.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to Geometry object.
<i>x</i>	on completion this variable will contain the Point X coordinate
<i>y</i>	on completion this variable will contain the Point Y coordinate

Returns

0 on failure: any other value on success

See also

[gaiaGetPointOnSurface](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.111 gaiaGetProjDatabasePath()

```
GAIAGEO_DECLARE const char* gaiaGetProjDatabasePath (
    const void * p_cache )
```

Return the currently set PATH leading to the private PROJ.6 database.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
----------------	--

Returns

the currently set PATH leading to the private PROJ.6 database; NULL if no PROJ database is defined.

See also

[gaiaSetProjDatabasePath](#)

Note

reentrant and thread-safe.

Remarks

PROJ.6 support required.

6.5.2.112 gaiaGetProjErrorMsg_r()

```
GAIA GEO_DECLARE const char* gaiaGetProjErrorMsg_r (  
    const void * p_cache )
```

Return the latest PROJ error message (if any)

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
----------------	--

Returns

the latest PROJ error message: an empty string if no error was previously found.

See also

[gaiaSetProjErrorMsg_r](#), [gaiaResetProjErrorMsg_r](#)

Note

reentrant and thread-safe.

Remarks

PROJ.6 support required.

6.5.2.113 gaiaGetProjString()

```
GAIA GEO_DECLARE char* gaiaGetProjString (
    const void * p_cache,
    const char * auth_name,
    int auth_srid )
```

Return the proj-string corresponding to a given CRS defined within the private PROJ.6 database.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>auth_name</i>	if NULL will be interpreted as "EPSG"
<i>auth_srid</i>	the intended CRS is identified by auth_name and auth_srid

Returns

the proj-string expression corresponding to the given CRS NULL on failite.

Note

you are responsible to destroy (before or after) any proj-string returned by [gaiaGetProjString\(\)](#) \nreentrant and thread-safe.

Remarks

PROJ.6 support required.

6.5.2.114 gaiaGetProjWKT()

```
GAIA GEO_DECLARE char* gaiaGetProjWKT (
    const void * p_cache,
    const char * auth_name,
    int auth_srid,
    int style,
    int indented,
    int indentation )
```

Return the WKT expression corresponding to a given CRS defined within the private PROJ.6 database.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>auth_name</i>	if NULL will be interpreted as "EPSG"
<i>auth_srid</i>	the intended CRS is identified by auth_name and auth_srid
<i>style</i>	one between GAIA_PROJ_WKT_GDAL, GAIA_PROJ_WKT_ESRI, GAIA_PROJ_WKT_ISO_2015 or GAIA_PROJ_WKT_ISO_2018 (this latter being the default assumption).
<i>indented</i>	if TRUE the WKT expression will be properly indented, otherwise a solid string lacking any white space or new-line will be printed.
<i>indentation</i>	how many chars to be used for indenting (only considered if indented is TRUE).

Returns

the WKT expression corresponding to the given CRS NULL on failite.

Note

you are responsible to destroy (before or after) any WKT string returned by [gaiaGetProjWKT\(\)](#) \nreentrant and thread-safe.

Remarks

PROJ.6 support required.

6.5.2.115 gaiaGetRtTopoErrorMsg()

```
GAIAGEO_DECLARE const char* gaiaGetRtTopoErrorMsg (  
    const void * p_cache )
```

Return the latest RTTOPO error message (if any)

Returns

the latest RTTOPO error message: an empty string if no error was previoysly found.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
----------------	--

See also

[gaiaResetRtTopoMsg](#), [gaiaGetRtTopoWarningMsg](#), [gaiaSetRtTopoErrorMsg](#), [gaiaSetRtTopoWarningMsg](#)

Remarks

RTTOPO support required.

6.5.2.116 gaiaGetRtTopoWarningMsg()

```
GAIAGEO_DECLARE const char* gaiaGetRtTopoWarningMsg (  
    const void * p_cache )
```

Return the latest RTTOPO warning message (if any)

Returns

the latest RTTOPO warning message: an empty string if no warning was previoysly found.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
----------------	--

See also

[gaiaResetRtTopoMsg](#), [gaiaGetRtTopoErrorMsg](#), [gaiaSetRtTopoErrorMsg](#), [gaiaSetRtTopoWarningMsg](#)

Note

not reentrant and thread unsafe.

Remarks

RTTOPO support required.

6.5.2.117 gaiaGuessSridFromWKT()

```
GAIA_GEO_DECLARE int gaiaGuessSridFromWKT (
    sqlite3 * db_handle,
    const void * p_cache,
    const char * wkt,
    int * srid )
```

Return the SRID corresponding to a given WKT expression (if any)

Parameters

<i>db_handle</i>	pointer to the current DB connection.
<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>wkt</i>	the WKT expression to be evaluated.
<i>srid</i>	on successful completion will point to the SRID value matching the WKT expression.

Returns

0 on failure: any other value on success

Remarks

PROJ.6 support required.

6.5.2.118 `gaiaHausdorffDistance()`

```
GAIAGEO_DECLARE int gaiaHausdorffDistance (
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2,
    double * dist )
```

Calculates the Hausdorff distance intercurring between two Geometry objects.

Parameters

<i>geom1</i>	pointer to first Geometry object
<i>geom2</i>	pointer to second Geometry object
<i>dist</i>	on completion this variable will contain the calculated Hausdorff distance

Returns

0 on failure: any other value on success.

See also

[gaiaHausdorffDistance_r](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS-ADVANCED support required.

6.5.2.119 `gaiaHausdorffDistance_r()`

```
GAIAGEO_DECLARE int gaiaHausdorffDistance_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2,
    double * dist )
```

Calculates the Hausdorff distance intercurring between two Geometry objects.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	pointer to first Geometry object
<i>geom2</i>	pointer to second Geometry object
<i>dist</i>	on completion this variable will contain the calculated Hausdorff distance

Returns

0 on failure: any other value on success.

See also

[gaiaHausdorffDistance](#)

Note

reentrant and thread-safe.

Remarks

GEOS-ADVANCED support required.

6.5.2.120 [gaiaHausdorffDistanceDensify\(\)](#)

```
GAIA GEO_DECLARE int gaiaHausdorffDistanceDensify (  
    gaiaGeomCollPtr geom1,  
    gaiaGeomCollPtr geom2,  
    double densify_fract,  
    double * dist )
```

Calculates the Hausdorff distance intercurring between two Geometry objects.

Parameters

<i>geom1</i>	pointer to first Geometry object
<i>geom2</i>	pointer to second Geometry object
<i>densify_fract</i>	fraction (in the range 0.0 / 1.0) by which to densify each segment. Each segment will be split into a number of equal-length subsegments, whose fraction of the total length is closest to the given fraction
<i>dist</i>	on completion this variable will contain the calculated Hausdorff distance

Returns

0 on failure: any other value on success.

See also

[gaiaHausdorffDistance_r](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS-ADVANCED support required.

6.5.2.121 `gaiaHausdorffDistanceDensify_r()`

```
GAIAGEO_DECLARE int gaiaHausdorffDistanceDensify_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2,
    double densify_fract,
    double * dist )
```

Calculates the Hausdorff distance intercurring between two Geometry objects.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	pointer to first Geometry object
<i>geom2</i>	pointer to second Geometry object
<i>densify_fract</i>	fraction (in the range 0.0 / 1.0) by which to densify each segment. Each segment will be split into a number of equal-lenght subsegments, whose fraction of the total length is closest to the given fraction
<i>dist</i>	on completion this variable will contain the calculated Hausdorff distance

Returns

0 on failure: any other value on success.

See also

[gaiaHausdorffDistance](#)

Note

reentrant and thread-safe.

Remarks

GEOS-ADVANCED support required.

6.5.2.122 `gaiaHexagonalGrid()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaHexagonalGrid (
    gaiaGeomCollPtr geom,
    double origin_x,
    double origin_y,
    double size,
    int mode )
```

Utility function: HexagonalGrid.

Parameters

<i>geom</i>	the Geometry to be covered by the Grid.
<i>origin</i> ↔ <i>_x</i>	the X ccoordinate identifying the Grid Origin.
<i>origin</i> ↔ <i>_y</i>	the Y coordinate identifying the Grid Origin.
<i>size</i>	the Grid cell-side size.
<i>mode</i>	any positive value will return a MULTILINESTRING, any will return a MULTIPOINT; zero will return a MULTIPOLYGON containing hexagonal POLYGONS.

Returns

the pointer to newly created Geometry object: NULL on failure.
 this function will always return a MultiPolygon
 NULL will be returned if any argument is invalid.

See also

[gaiaGexagonalGrid_r](#), [gaiaFreeGeomColl](#), [gaiaSquareGrid](#), [gaiaTriangularGrid](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaHexagonalGrid\(\)](#)
 not reentrant and thread unsafe.

6.5.2.123 [gaiaHexagonalGrid_r\(\)](#)

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaHexagonalGrid\_r (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    double origin_x,
    double origin_y,
    double size,
    int mode )
```

Utility function: HexagonalGrid.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	the Geometry to be covered by the Grid.
<i>origin</i> ↔ <i>_x</i>	the X ccoordinate identifying the Grid Origin.
<i>origin</i> ↔ <i>_y</i>	the Y coordinate identifying the Grid Origin.
<i>size</i>	the Grid cell-side size.
<i>mode</i>	any positive value will return a MULTILINESTRING, any will return a MULTIPOINT; zero will return a MULTIPOLYGON containing hexagonal POLYGONS.

Returns

the pointer to newly created Geometry object: NULL on failure.
 this function will always return a MultiPolygon
 NULL will be returned if any argument is invalid.

See also

[gaiaGexagonalGrid](#), [gaiaFreeGeomColl](#), [gaiaSquareGrid](#), [gaiaTriangularGrid](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaHexagonalGrid_r\(\)](#)
 reentrant and thread-safe.

6.5.2.124 gaiaHilbertCode()

```
GAIA_GEO_DECLARE int gaiaHilbertCode (
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2,
    int level,
    unsigned int * code )
```

Hilbert Code.

Parameters

<i>geom1</i>	the input Geometry object.
<i>geom2</i>	the Extent
<i>level</i>	integer precision level of the Hilbert Curve [1-16]
<i>code</i>	pointer to an unsigned int where to store the calculated Hilbert Code

Returns

1 on success, 0 on failure.

See also

[gaiaHilbertCode_r](#)

Remarks

GEOS_3110 support required.

6.5.2.125 `gaiaHilbertCode_r()`

```
GAIAGEO_DECLARE int gaiaHilbertCode_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2,
    int level,
    unsigned int * code )
```

Hilbert Code.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the input Geometry object.
<i>geom2</i>	the Extent
<i>level</i>	integer precision level of the Hilbert Curve [1-16]
<i>code</i>	pointer to an unsigned int where to store the calculated Hilbert Code

Returns

1 on success, 0 on failure.

See also

[gaiaHilbertCode](#)

Remarks

GEOS_3110 support required.

6.5.2.126 `gaiaIntersectionMatrixPatternMatch()`

```
GAIAGEO_DECLARE int gaiaIntersectionMatrixPatternMatch (
    const char * matrix,
    const char * pattern )
```

Spatial relationship evaluation: comparing two intersection matrices.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>matrix</i>	first intersection matrix [DE-9IM] to be compared
<i>pattern</i>	second intersection matrix [DE-9IM] to be compared (reference pattern)

Returns

0 if false: any other value if true; -1 on invalid args

See also

[gaiaGeomCollRelateBoundaryNodeRule](#), [gaiaIntersectionMatrixPatternMatch_r](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.127 gaiaIntersectionMatrixPatternMatch_r()

```
GAIA GEO_DECLARE int gaiaIntersectionMatrixPatternMatch_r (
    const void * p_cache,
    const char * matrix,
    const char * pattern )
```

Spatial relationship evaluation: comparing two intersection matrices.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>matrix</i>	first intersection matrix [DE-9IM] to be compared
<i>pattern</i>	second intersection matrix [DE-9IM] to be compared (reference pattern)

Returns

0 if false: any other value if true; -1 on invalid args

See also

[gaiaGeomCollRelateBoundaryNodeRule](#), [gaiaIntersectionMatrixPatternMatch](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.128 `gaialsClosed()`

```
GAIA GEO_DECLARE int gaiaIsClosed (
    gaiaLinestringPtr line )
```

Checks if a Linestring object represents an OGC Closed Geometry.

This function only works on a single linestring - if you pass in a multi-line linestring geometry, it will return 0 (false). See `gaialsClosedGeom` for an alternative.

Parameters

<i>line</i>	pointer to Linestring object.
-------------	-------------------------------

Returns

0 if false; any other value if true

See also

[gaialsSimple](#), [gaialsRing](#), [gaialsValid](#), [gaialsClosedGeom](#)

Remarks

GEOS support required.

6.5.2.129 `gaialsClosedGeom()`

```
GAIA GEO_DECLARE int gaiaIsClosedGeom (
    gaiaGeomCollPtr geom )
```

Checks if a Geometry object represents an OGC Closed Linestring.

Parameters

<i>geom</i>	pointer to Geometry object.
-------------	-----------------------------

Returns

0 if false; any other value if true

See also

[gaialsClosedGeom_r](#), [gaialsSimple](#), [gaialsRing](#), [gaialsValid](#), [gaialsClosed](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.130 gaiaIsClosedGeom_r()

```
GAIAGEO_DECLARE int gaiaIsClosedGeom_r (
    const void * p_cache,
    gaiaGeomCollPtr geom )
```

Checks if a Geometry object represents an OGC Closed Linestring.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to Geometry object.

Returns

0 if false; any other value if true

See also

[gaiaIsClosedGeom](#), [gaiaIsSimple](#), [gaiaIsRing](#), [gaiaIsValid](#), [gaiaIsClosed](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.131 gaiaIsRing()

```
GAIAGEO_DECLARE int gaiaIsRing (
    gaiaLinestringPtr line )
```

Checks if a Linestring object represents an OGC Ring Geometry.

Parameters

<i>line</i>	pointer to Geometry object.
-------------	-----------------------------

Returns

0 if false; any other value if true

See also

[gaialsRing_r](#), [gaialsSimple](#), [gaialsClosed](#), [gaialsValid](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.132 gaialsRing_r()

```
GAIAGEO_DECLARE int gaiaIsRing_r (  
    const void * p_cache,  
    gaiaLinestringPtr line )
```

Checks if a Linestring object represents an OGC Ring Geometry.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>line</i>	pointer to Geometry object.

Returns

0 if false; any other value if true

See also

[gaialsRing](#), [gaialsSimple](#), [gaialsClosed](#), [gaialsValid](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.133 `gaialsSimple()`

```
GAIA GEO_DECLARE int gaiaIsSimple (
    gaiaGeomCollPtr geom )
```

Checks if a Geometry object represents an OGC Simple Geometry.

Parameters

<i>geom</i>	pointer to Geometry object.
-------------	-----------------------------

Returns

0 if false; any other value if true

See also

[gaialsSimple_r](#), [gaialsClosed](#), [gaialsRing](#), [gaialsValid](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.134 `gaialsSimple_r()`

```
GAIA GEO_DECLARE int gaiaIsSimple_r (
    const void * p_cache,
    gaiaGeomCollPtr geom )
```

Checks if a Geometry object represents an OGC Simple Geometry.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to Geometry object.

Returns

0 if false; any other value if true

See also

[gaialsSimple](#), [gaialsClosed](#), [gaialsRing](#), [gaialsValid](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.135 gaialsValid()

```
GAIAGEO_DECLARE int gaiaIsValid (
    gaiaGeomCollPtr geom )
```

Checks if a Geometry object represents an OGC Valid Geometry.

Parameters

<i>geom</i>	pointer to Geometry object.
-------------	-----------------------------

Returns

0 if false; any other value if true

See also

[gaialsValid_r](#), [gaialsSimple](#), [gaialsClosed](#), [gaialsRing](#), [gaialsValidReason](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.136 gaialsValid_r()

```
GAIAGEO_DECLARE int gaiaIsValid_r (
    const void * p_cache,
    gaiaGeomCollPtr geom )
```

Checks if a Geometry object represents an OGC Valid Geometry.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to Geometry object.

Returns

0 if false; any other value if true

See also

[gaialsValid](#), [gaialsSimple](#), [gaialsClosed](#), [gaialsRing](#), [gaialsValidReason_r](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

Examples

[demo2.c](#).

6.5.2.137 gaialsValidDetail()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaIsValidDetail (
    gaiaGeomCollPtr geom )
```

return a Geometry detail causing a Geometry to be invalid

Parameters

<i>geom</i>	pointer to the Geometry object to be validated.
-------------	---

Returns

pointer to a Geometry object causing invalidity, or NULL.

See also

[gaialsValid](#), [gaialsValidReason](#), [gaialsValidDetail_r](#)

Note

you are responsible to destroy the returned Geometry
not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.138 gaiaIsValidDetail_r()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaIsValidDetail_r (
    const void * p_cache,
    gaiaGeomCollPtr geom )
```

return a Geometry detail causing a Geometry to be invalid

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to the Geometry object to be validated.

Returns

pointer to a Geometry object causing invalidity, or NULL.

See also

[gaiaIsValid_r](#), [gaiaIsValidReason_r](#), [gaiaIsValidDetail](#)

Note

you are responsible to destroy the returned Geometry
reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.139 gaiaIsValidDetailEx()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaIsValidDetailEx (
    gaiaGeomCollPtr geom,
    int esri_flag )
```

return a Geometry detail causing a Geometry to be invalid

Parameters

<i>geom</i>	pointer to the Geometry object to be validated.
<i>esri_flag</i>	if set to TRUE if set to TRUE all ESRI-like holes (violating the basic OGC model) will be considered to be valid.

Returns

pointer to a Geometry object causing invalidity, or NULL.

See also

[gaialsValid](#), [gaialsValidReason](#), [gaialsValidDetail_r](#)

Note

you are responsible to destroy the returned Geometry
not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.140 gaialsValidDetailEx_r()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaIsValidDetailEx_r (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    int esri_flag )
```

return a Geometry detail causing a Geometry to be invalid

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to the Geometry object to be validated.
<i>esri_flag</i>	if set to TRUE all ESRI-like holes (violating the basic OGC model) will be considered to be valid.

Returns

pointer to a Geometry object causing invalidity, or NULL.

See also

[gaialsValid_r](#), [gaialsValidReason_r](#), [gaialsValidDetail](#)

Note

you are responsible to destroy the returned Geometry
reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.141 gaialsValidReason()

```
GAIA GEO_DECLARE char* gaiaIsValidReason (
    gaiaGeomCollPtr geom )
```

return a TEXT string stating if a Geometry is valid and if not valid, a reason why

Parameters

<i>geom</i>	pointer to the Geometry object to be validated.
-------------	---

Returns

a text string.

See also

[gaialsValid](#), [gaialsValidReason_r](#), [gaialsValidDetail](#)

Note

you are responsible to free() the returned text string
not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.142 gaialsValidReason_r()

```
GAIA GEO_DECLARE char* gaiaIsValidReason_r (
    const void * p_cache,
    gaiaGeomCollPtr geom )
```

return a TEXT string stating if a Geometry is valid and if not valid, a reason why

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to the Geometry object to be validated.

Returns

a text string.

See also

[gaialsValid_r](#), [gaialsValidReason](#), [gaialsValidDetail_r](#)

Note

you are responsible to free() the returned text string
reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.143 gaiaLargestEmptyCircle()

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaLargestEmptyCircle (
    gaiaGeomCollPtr geom,
    gaiaGeomCollPtr boundary,
    double tolerance )
```

Constructs the Largest Empty Circle for a set of obstacle geometries, up to a specified tolerance.

Parameters

<i>geom</i>	pointer to input Geometry object
<i>tolerance</i>	approximation factor

Returns

NULL on failure: the maximum inscribed circle on success.

See also

[gaiaLargestEmptyCircle_r](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS-ADVANCED support required.

6.5.2.144 gaiaLargestEmptyCircle_r()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLargestEmptyCircle_r (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    gaiaGeomCollPtr boundary,
    double tolerance )
```

Constructs the Largest Empty Circle for a set of obstacle geometries, up to a specified tolerance.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to input Geometry object
<i>tolerance</i>	approximation factor

Returns

NULL on failure: the maximum inscribed circle on success.

See also

[gaiaLargestEmptyCircle](#)

Note

reentrant and thread-safe.

Remarks

GEOS-ADVANCED support required.

6.5.2.145 gaiaLineFromEncodedPolyline()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLineFromEncodedPolyline (
    const void * p_cache,
    const char * encoded,
    unsigned char precision )
```

Converts a GoogleMaps encoded PolyLine into a native binary Geometry (Linestring)

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>encoded</i>	pointer to GoogleMaps encoded PolyLine
<i>precision</i>	how much precision (decimal digits)

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaAsEncodedPolyLine](#)

Note

you are responsible to destroy the native geometry returned by [gaiaFromTWKB\(\)](#).

Remarks

RTTOPO support required.

6.5.2.146 [gaiaLineInterpolateEquidistantPoints\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLineInterpolateEquidistantPoints (
    gaiaGeomCollPtr in_geom,
    double distance )
```

Spatial operator: Line Interpolate Equidistant Points.

Parameters

<i>in_geom</i>	the input Geometry object [expected to be of lineal type]
<i>distance</i>	regular distance between interpolated points

Returns

the pointer to newly created Geometry object representing a MultiPoint; such MultiPoint always supports the M coordinate (the corresponding value representing the progressive distance for each interpolated Point). individual Points will be regularly spaced by the given distance: NULL on failure.

See also

[gaiaLineInterpolateEquidistantPoints_r](#), [gaiaLineInterpolatePoint](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaLineInterpolateEquidistantPoints\(\)](#)
not reentrant and thread unsafe.

Remarks

GEOS-ADVANCED support required.

6.5.2.147 gaiaLineInterpolateEquidistantPoints_r()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLineInterpolateEquidistantPoints_r (
    const void * p_cache,
    gaiaGeomCollPtr ln_geom,
    double distance )
```

Spatial operator: Line Interpolate Equidistant Points.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>ln_geom</i>	the input Geometry object [expected to be of lineal type]
<i>distance</i>	regular distance between interpolated points

Returns

the pointer to newly created Geometry object representing a MultiPoint; such MultiPoint always supports the M coordinate (the corresponding value representing the progressive distance for each interpolated Point). individual Points will be regularly spaced by the given distance: NULL on failure.

See also

[gaiaLineInterpolateEquidistantPoints](#), [gaiaLineInterpolatePoint](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaLineInterpolateEquidistantPoints_r\(\)](#)
reentrant and thread-safe.

Remarks

GEOS-ADVANCED support required.

6.5.2.148 gaiaLineInterpolatePoint()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLineInterpolatePoint (
    gaiaGeomCollPtr ln_geom,
    double fraction )
```

Spatial operator: Line Interpolate Point.

Parameters

<i>ln_geom</i>	the input Geometry object [expected to be of lineal type]
<i>fraction</i>	total length fraction [in the range 0.0 / 1.0]

Returns

the pointer to newly created Geometry object representing a Point laying on the input Geometry and positioned at the given length fraction: NULL on failure.

See also

[gaiaLineInterpolatePoint_r](#), [gaiaLineInterpolateEquidistantPoints](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaLineInterpolatePoint\(\)](#)
not reentrant and thread unsafe.

Remarks

GEOS-ADVANCED support required.

6.5.2.149 [gaiaLineInterpolatePoint_r\(\)](#)

```
GAIGEO_DECLARE gaiaGeomCollPtr gaiaLineInterpolatePoint_r (
    const void * p_cache,
    gaiaGeomCollPtr ln_geom,
    double fraction )
```

Spatial operator: Line Interpolate Point.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>ln_geom</i>	the input Geometry object [expected to be of lineal type]
<i>fraction</i>	total length fraction [in the range 0.0 / 1.0]

Returns

the pointer to newly created Geometry object representing a Point laying on the input Geometry and positioned at the given length fraction: NULL on failure.

See also

[gaiaLineInterpolatePoint](#), [gaiaLineInterpolateEquidistantPoints](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaLineInterpolatePoint_r\(\)](#) reentrant and thread-safe.

Remarks

GEOS-ADVANCED support required.

6.5.2.150 gaiaLineLocatePoint()

```
GAIAGEO_DECLARE double gaiaLineLocatePoint (
    gaiaGeomCollPtr ln_geom,
    gaiaGeomCollPtr pt_geom )
```

Determines the location of the closest Point on Linestring to the given Point.

Parameters

<i>ln_geom</i>	pointer to first input Geometry object [expected to be of the lineal type].
<i>pt_geom</i>	pointer to second input Geometry object [expected to be a Point].

Returns

the fraction [in the range 0.0 / 1.0] of *ln_geom* total length where the closest Point to *pt_geom* lays.

See also

[gaiaLineLocatePoint_r](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS-ADVANCED support required.

6.5.2.151 gaiaLineLocatePoint_r()

```
GAIAGEO_DECLARE double gaiaLineLocatePoint_r (
    const void * p_cache,
    gaiaGeomCollPtr ln_geom,
    gaiaGeomCollPtr pt_geom )
```

Determines the location of the closest Point on Linestring to the given Point.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>ln_geom</i>	pointer to first input Geometry object [expected to be of the lineal type].
<i>pt_geom</i>	pointer to second input Geometry object [expected to be a Point].

Returns

the fraction [in the range 0.0 / 1.0] of *ln_geom* total length where the closest Point to *pt_geom* lays.

See also

[gaiaLineLocatePoint](#)

Note

reentrant and thread-safe.

Remarks

GEOS-ADVANCED support required.

6.5.2.152 **gaiaLineMerge()**

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaLineMerge (
    gaiaGeomCollPtr geom )
```

Spatial operator: Line Merge.

Parameters

<i>geom</i>	pointer to input Geometry object.
-------------	-----------------------------------

Returns

the pointer to newly created Geometry object; NULL on failure.
if possible, this representing a reassembled Linestring or MultiLinestring.

See also

[gaiaLineMerge_r](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaLineMerge\(\)](#)
not reentrant and thread unsafe.

Remarks

GEOS-ADVANCED support required.

6.5.2.153 `gaiaLineMerge_r()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLineMerge_r (
    const void * p_cache,
    gaiaGeomCollPtr geom )
```

Spatial operator: Line Merge.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to input Geometry object.

Returns

the pointer to newly created Geometry object; NULL on failure.
if possible, this representing a reassembled Linestring or MultiLinestring.

See also

[gaiaLineMerge](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaLineMerge_r\(\)](#)
reentrant and thread-safe.

Remarks

GEOS-ADVANCED support required.

6.5.2.154 `gaiaLinesCutAtNodes()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLinesCutAtNodes (
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial operator: Line Cut At Nodes.

Parameters

<i>geom1</i>	pointer to input Geometry object [Linestring or MultiLinestring].
<i>geom2</i>	pointer to input Geometry object [Point or MultiPoint].

Returns

the pointer to newly created Geometry object; NULL on failure.
 if possible, any input Linestring will be split accordingly to given Node(s): no point will be interpolated, existing Linestring Vertices will be evaluated.

See also

[gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaLinesCutAtNodes\(\)](#)

Remarks

GEOS-ADVANCED support required.

6.5.2.155 **gaiaLineSubstring()**

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaLineSubstring (
    gaiaGeomCollPtr ln_geom,
    double start_fraction,
    double end_fraction )
```

Spatial operator: Line Substring.

Parameters

<i>ln_geom</i>	the input Geometry object [expected to be of lineal type]
<i>start_fraction</i>	substring start, expressed as total length fraction [in the range 0.0 / 1.0]
<i>end_fraction</i>	substring end, expressed as total length fraction

Returns

the pointer to newly created Geometry object representing a Linestring laying on the input Geometry.
 this Linestring will begin (and stop) at given total length fractions. NULL on failure.

See also

[gaiaLineSubstring_r](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaLineSubstring\(\)](#)
not reentrant and thread unsafe.

Remarks

GEOS-ADVANCED support required.

6.5.2.156 [gaiaLineSubstring_r\(\)](#)

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaLineSubstring\_r (  
    const void * p_cache,  
    gaiaGeomCollPtr ln_geom,  
    double start_fraction,  
    double end_fraction )
```

Spatial operator: Line Substring.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>ln_geom</i>	the input Geometry object [expected to be of lineal type]
<i>start_fraction</i>	substring start, expressed as total length fraction [in the range 0.0 / 1.0]
<i>end_fraction</i>	substring end, expressed as total length fraction

Returns

the pointer to newly created Geometry object representing a Linestring laying on the input Geometry.
this Linestring will begin (and stop) at given total length fractions. NULL on failure.

See also

[gaiaLineSubstring](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaLineSubstring_r\(\)](#)
reentrant and thread-safe.

Remarks

GEOS-ADVANCED support required.

6.5.2.157 gaiaMakeValid()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaMakeValid (
    const void * p_cache,
    gaiaGeomCollPtr geom )
```

Utility function: MakeValid.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	the input Geometry object.

Returns

the pointer to newly created Geometry object: NULL on failure.
 this function will attempt to create a valid representation of a given invalid geometry without losing any of the input vertices.
 Already-valid geometries are returned without further intervention.
 NULL will be returned if the passed argument is invalid.

See also

[gaiaFreeGeomColl](#), [gaiaMakeValidDiscarded](#), [gaiaGeosMakeValid](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaMakeValid\(\)](#)

Remarks

RTTOPO support required.

6.5.2.158 gaiaMakeValidDiscarded()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaMakeValidDiscarded (
    const void * p_cache,
    gaiaGeomCollPtr geom )
```

Utility function: MakeValidDiscarded.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	the input Geometry object.

Returns

the pointer to newly created Geometry object: NULL on failure.
 this function will attempt to collect any invalid item (offending geometries) discarded by `gaiaMakeValid` while building a valid Geometry.
 Saving any discarded item could be useful for a finer (manual) adjustment.
 NULL will be returned if `gaiaMakeValid` hasn't identified any offending item to be discarded during the validation.

See also

[gaiaFreeGeomColl](#), [gaiaMakeValid](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaMakeValidDiscarded\(\)](#)

Remarks

RTTOPO support required.

6.5.2.159 gaiaMaxDistance()

```
GAIA GEO_DECLARE int gaiaMaxDistance (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2,
    double * dist )
```

Calculates the maximum 2D distance intercurring between two Geometry objects.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry object
<i>geom2</i>	the second Geometry object
<i>dist</i>	on completion this variable will contain the calculated distance

Returns

0 on failure: any other value on success.

See also

[gaiaGeomCollDistance](#), [gaia3DDistance](#), [gaia3DMaxDistance](#)

Note

this function computes the 2D maximum cartesian distance (Z is always ignored)

Remarks

RTTOPO support required.

6.5.2.160 gaiaMaximumInscribedCircle()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaMaximumInscribedCircle (
    gaiaGeomCollPtr geom,
    double tolerance )
```

Constructs the Maximum Inscribed Circle for a polygonal geometry, up to a specified tolerance.

Parameters

<i>geom</i>	pointer to input Geometry object
<i>tolerance</i>	approximation factor

Returns

NULL on failure: the maximum inscribed circle on success.

See also

[gaiaMaximumInscribedCircle_r](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS-ADVANCED support required.

6.5.2.161 gaiaMaximumInscribedCircle_r()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaMaximumInscribedCircle_r (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    double tolerance )
```

Constructs the Maximum Inscribed Circle for a polygonal geometry, up to a specified tolerance.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to input Geometry object
<i>tolerance</i>	approximation factor

Returns

0 on failure: the maximum inscribed circle on success.

See also

[gaiaMaximumInscribedCircle](#)

Note

reentrant and thread-safe.

Remarks

GEOS-ADVANCED support required.

6.5.2.162 [gaiaMinimumBoundingCircle\(\)](#)

```
GAIA_GEO_DECLARE gaiaGeomCollPtr gaiaMinimumBoundingCircle (  
    gaiaGeomCollPtr geom,  
    double * radius,  
    gaiaGeomCollPtr * center )
```

Constructs the Minimum Bounding Circle for a generic geometry.

Parameters

<i>geom</i>	pointer to input Geometry object
<i>radius</i>	on completion this variable will contain the radius of the circle
<i>center</i>	on completion this variable will contain the centre of the circle (POINT Geometry)

Returns

NULL on failure: the minimum bounding circle on success.

See also

[gaiaMinimumBoundingCircle_r](#)

Note

not reentrant and thread unsafe.

you are responsible to destroy (before or after) any allocated Geometry, this including the Geometry returned by `gaiaMinumumBoundingCircle()` and the POINT Geometry corresponding to the Center

Remarks

GEOS-ADVANCED support required.

6.5.2.163 `gaiaMinimumBoundingCircle_r()`

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaMinimumBoundingCircle_r (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    double * radius,
    gaiaGeomCollPtr * center )
```

Constructs the Minimum Bounding Circle for a generic geometry.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to input Geometry object
<i>radius</i>	on completion this variable will contain the radius of the circle
<i>center</i>	on completion this variable will contain the centre of the circle (POINT Geometry)

Returns

NULL on failure: the minimum bounding circle on success.

See also

[gaiaMinimumBoundingCircle](#)

Note

reentrant and thread-safe.

you are responsible to destroy (before or after) any allocated Geometry, this including the Geometry returned by `gaiaMinumumBoundingCircle()` and the POINT Geometry corresponding to the Center

Remarks

GEOS-ADVANCED support required.

6.5.2.164 `gaiaMinimumClearance()`

```
GAIA GEO_DECLARE int gaiaMinimumClearance (
    gaiaGeomCollPtr geom,
    double * clearance )
```

Computes the minimum clearance of a geometry.

Parameters

<i>geom</i>	pointer to input Geometry object
<i>clearance</i>	on succesfull completion this pointer will reference the computed clearance value

Returns

ZERO (0) on failure: any other value on success

See also

[gaiaMinimumClearance_r](#), [gaiaMinimumClearanceLine](#), [gaiaMinimumClearanceLine_r](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS-ADVANCED support required.

6.5.2.165 `gaiaMinimumClearance_r()`

```
GAIA GEO_DECLARE int gaiaMinimumClearance_r (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    double * clearance )
```

Computes the minimum clearance of a geometry.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to input Geometry object
<i>clearance</i>	on succesfull completion this pointer will reference the computed clearance value

Returns

ZERO (0) on failure: any other value on success

See also

[gaiaMinimumClearance_r](#), [gaiaMinimumClearanceLine](#), [gaiaMinimumClearanceLine_r](#)

Note

reentrant and thread-safe.

Remarks

GEOS-ADVANCED support required.

6.5.2.166 gaiaMinimumClearanceLine()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaMinimumClearanceLine (
    gaiaGeomCollPtr geom )
```

Returns a LINESTRING geometry which represents the minimum clearance of a geometry.

Parameters

<i>geom</i>	pointer to input Geometry object
-------------	----------------------------------

Returns

NULL on failure: the minimum clearance Linestring on success.

See also

[gaiaMinimumClearance](#), [gaiaMinimumClearance_r](#), [gaiaMinimumClearanceLine_r](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS-ADVANCED support required.

6.5.2.167 gaiaMinimumClearanceLine_r()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaMinimumClearanceLine_r (
    const void * p_cache,
    gaiaGeomCollPtr geom )
```

Returns a LINESTRING geometry which represents the minimum clearance of a geometry.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to input Geometry object

Returns

NULL on failure: the minimum clearance Linestring on success.

See also

[gaiaMinimumClearance](#), [gaiaMinimumClearance_r](#), [gaiaMinimumClearanceLine](#)

Note

reentrant and thread-safe.

Remarks

GEOS-ADVANCED support required.

6.5.2.168 [gaiaMinimumRotatedRectangle\(\)](#)

```
GAIA_GEO_DECLARE gaiaGeomCollPtr gaiaMinimumRotatedRectangle (  
    gaiaGeomCollPtr geom )
```

Calculates the minimum rotated rectangular POLYGON which encloses the input geometry.

Parameters

<i>geom</i>	pointer to input Geometry object
-------------	----------------------------------

Returns

NULL on failure: the minimum rotated rectangle on success.

See also

[gaiaMinimumRotatedRectangle_r](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS-ADVANCED support required.

6.5.2.169 `gaiaMinimumRotatedRectangle_r()`

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaMinimumRotatedRectangle_r (
    const void * p_cache,
    gaiaGeomCollPtr geom )
```

Calculates the minimum rotated rectangular POLYGON which encloses the input geometry.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to input Geometry object

Returns

NULL on failure: the minimum rotated rectangle on success.

See also

[gaiaMinimumRotatedRectangle](#)

Note

reentrant and thread-safe.

Remarks

GEOS-ADVANCED support required.

6.5.2.170 `gaiaMinimumWidth()`

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaMinimumWidth (
    gaiaGeomCollPtr geom )
```

Returns a LINESTRING geometry which represents the minimum diameter of the geometry.

Parameters

<i>geom</i>	pointer to input Geometry object
-------------	----------------------------------

Returns

NULL on failure: the minimum width Linestring on success.

See also

[gaiaMinimumWidth_r](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS-ADVANCED support required.

6.5.2.171 [gaiaMinimumWidth_r\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaMinimumWidth\_r (  
    const void * p_cache,  
    gaiaGeomCollPtr geom )
```

Returns a LINESTRING geometry which represents the minimum diameter of the geometry.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to input Geometry object

Returns

NULL on failure: the minimum width Linestring on success.

See also

[gaiaMinimumWidth](#)

Note

reentrant and thread-safe.

Remarks

GEOS-ADVANCED support required.

6.5.2.172 [gaiaNodeLines\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaNodeLines (  
    const void * p_cache,  
    gaiaGeomCollPtr input )
```

Utility function: re-noding lines.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>input</i>	the input Geometry object.

Returns

the pointer to newly created Geometry object: NULL on failure.
 The function fully nodes a set of linestrings, using the least nodes preserving all the input ones.

See also

[gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaNodeLines\(\)](#)

Remarks

RTTOPO support required.

6.5.2.173 **gaiaOffsetCurve()**

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaOffsetCurve (
    gaiaGeomCollPtr geom,
    double radius,
    int points,
    int left_right )
```

Spatial operator: Offset Curve.

Parameters

<i>geom</i>	the input Geometry object
<i>radius</i>	the buffer's radius
<i>points</i>	number of points (aka vertices) to be used in order to approximate a circular arc.
<i>left_right</i>	if set to 1 the left-sided OffsetCurve will be returned; otherwise the right-sided one.

Returns

the pointer to newly created Geometry object representing the OffsetCurve of input Geometry: NULL on failure.

See also

[gaiaOffsetCurve_r](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaOffsetCurve\(\)](#)
not reentrant and thread unsafe.

Remarks

GEOS-ADVANCED support required.

6.5.2.174 [gaiaOffsetCurve_r\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaOffsetCurve_r (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    double radius,
    int points,
    int left_right )
```

Spatial operator: Offset Curve.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	the input Geometry object
<i>radius</i>	the buffer's radius
<i>points</i>	number of points (aka vertices) to be used in order to approximate a circular arc.
<i>left_right</i>	if set to 1 the left-sided OffsetCurve will be returned; otherwise the right-sided one.

Returns

the pointer to newly created Geometry object representing the OffsetCurve of input Geometry: NULL on failure.

See also

[gaiaOffsetCurve](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaOffsetCurve_r\(\)](#)
reentrant and thread-safe.

Remarks

GEOS-ADVANCED support required.

6.5.2.175 `gaiaPolygonize()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaPolygonize (
    gaiaGeomCollPtr geom,
    int force_multi )
```

Attempts to rearrange a generic Geometry object into a Polygon or MultiPolygon.

Parameters

<i>geom</i>	the input Geometry object
<i>force_multi</i>	if not set to 0, then an eventual Polygon will be returned casted to MultiPolygon

Returns

the pointer to newly created Geometry object representing a Polygon or MultiPolygon Geometry: NULL on failure.

See also

[gaiaPolygonize_r](#), [gaiaMakePolygon](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaPolygonize\(\)](#)
not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.176 `gaiaPolygonize_r()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaPolygonize_r (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    int force_multi )
```

Attempts to rearrange a generic Geometry object into a Polygon or MultiPolygon.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	the input Geometry object
<i>force_multi</i>	if not set to 0, then an eventual Polygon will be returned casted to MultiPolygon

Returns

the pointer to newly created Geometry object representing a Polygon or MultiPolygon Geometry: NULL on failure.

See also

[gaiaPolygonize](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaPolygonize_r\(\)](#)
reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.177 gaiaProjectedPoint()

```
GAIAGEO_DECLARE int gaiaProjectedPoint (
    const void * p_cache,
    double x1,
    double y1,
    double a,
    double b,
    double distance,
    double azimuth,
    double * x2,
    double * y2 )
```

Utility function: ProjectedPoint.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>x1</i>	the X coordinate of the Start Point.
<i>y1</i>	the Y coordinate of the Start Point.
<i>a</i>	major axis of the reference spheroid.
<i>b</i>	minor axis of the reference spheroid.
<i>distance</i>	a distance expressed in Meters
<i>azimuth</i>	(aka bearing aka heading) expressed in radians; on the clock: 12=0; 3=PI/2; 6=PI; 9=3PI/2.
<i>x2</i>	on completion this variable will contain the the X coordinate of the Projected Point.
<i>y2</i>	on completion this variable will contain the the Y coordinate of the Projected Point.

Returns

0 on failure: any other value on success

Remarks

RTTOPO support required.

6.5.2.178 gaiaRadsToDeps()

```
GAIAGEO_DECLARE double gaiaRadsToDeps (  
    double rads )
```

Converts and angle from Radians into Degrees.

Parameters

<i>rads</i>	the angle measured in Radians.
-------------	--------------------------------

Returns

the angle measured in Degrees.

See also

[gaiaDepsToRads](#)

Remarks

PROJ support required

6.5.2.179 gaiaReducePrecision()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaReducePrecision (  
    gaiaGeomCollPtr geom,  
    double grid_size )
```

Change the coordinate precision of a geometry.

Parameters

<i>geom</i>	the input Geometry object.
<i>grid_size</i>	the gridSize cell size of grid to round coordinates to, or 0 for FLOATING precision

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaFreeGeomColl](#), [gaiaReducePrecision_r](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaReducePrecision\(\)](#)

Remarks

GEOS_3100 support required.

6.5.2.180 [gaiaReducePrecision_r\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaReducePrecision\_r (  
    const void * p_cache,  
    gaiaGeomCollPtr geom,  
    double grid_size )
```

Change the coordinate precision of a geometry.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	the input Geometry object.
<i>grid_size</i>	the gridSize cell size of grid to round coordinates to, or 0 for FLOATING precision

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaFreeGeomColl](#), [gaiaReducePrecision](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaReducePrecision\(\)](#)

Remarks

GEOS_3100 support required.

6.5.2.181 `gaiaResetGeosMsg()`

```
GAIAGEO_DECLARE void gaiaResetGeosMsg (  
    void )
```

Resets the GEOS error and warning messages to an empty state.

See also

[gaiaResetGeosMsg_r](#), [gaiaGetGeosErrorMsg](#), [gaiaGetGeosWarningMsg](#), [gaiaGeosAuxErrorMsg](#), [gaiaSetGeosErrorMsg](#), [gaiaSetGeosWarningMsg](#), [gaiaSetGeosAuxErrorMsg](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.182 `gaiaResetGeosMsg_r()`

```
GAIAGEO_DECLARE void gaiaResetGeosMsg_r (  
    const void * p_cache )
```

Resets the GEOS error and warning messages to an empty state.

Parameters

<code><i>p_cache</i></code>	a memory pointer returned by spatialite_alloc_connection()
-----------------------------	--

See also

[gaiaResetGeosMsg](#), [gaiaGetGeosErrorMsg](#), [gaiaGetGeosWarningMsg](#), [gaiaGeosAuxErrorMsg](#), [gaiaSetGeosErrorMsg](#), [gaiaSetGeosWarningMsg](#), [gaiaSetGeosAuxErrorMsg](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.183 gaiaResetProjErrorMsg_r()

```
GAIA GEO_DECLARE void gaiaResetProjErrorMsg_r (  
    const void * p_cache )
```

Resets the PROJ error messages to an empty state.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
----------------	--

See also

[gaiaSetProjErrorMsg_r](#), [gaiaGetProjErrorMsg_r](#)

Note

reentrant and thread-safe.

Remarks

PROJ.6 support required.

6.5.2.184 **gaiaResetRtTopoMsg()**

```
GAIAGEO_DECLARE void gaiaResetRtTopoMsg (
    const void * p_cache )
```

Resets the RTTOPO error and warning messages to an empty state.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
----------------	--

See also

[gaiaGetRtTopoErrorMsg](#), [gaiaGetRtTopoWarningMsg](#), [gaiaSetRtTopoErrorMsg](#), [gaiaSetRtTopoWarningMsg](#)

Remarks

RTTOPO support required.

6.5.2.185 **gaiaSegmentize()**

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSegmentize (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    double dist )
```

Utility function: Segmentize.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	the input Geometry object.
<i>dist</i>	the maximum segment length.

Returns

the pointer to newly created Geometry object: NULL on failure.
this function will return a modified geometry having no segment longer than the given distance.
Distance computation is performed in 2d only.
all Points or segments shorter than 'dist' will be returned without further intervention.
NULL will be returned if the passed argument is invalid.

See also

[gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaSegmentize\(\)](#)

Remarks

RTTOPO support required.

6.5.2.186 [gaiaSetGeosAuxErrorMsg\(\)](#)

```
GAIA GEO_DECLARE void gaiaSetGeosAuxErrorMsg (  
    const char * msg )
```

Set the current GEOS (auxiliary) error message.

Parameters

<i>msg</i>	the error message to be set.
------------	------------------------------

See also

[gaiaSetAuxErrorMsg_r](#), [gaiaResetGeosMsg](#), [gaiaGetGeosErrorMsg](#), [gaiaGetGeosWarningMsg](#), [gaiaGetGeosAuxErrorMsg](#), [gaiaSetGeosWarningMsg](#), [gaiaSetGeosErrorMsg](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.187 `gaiaSetGeosAuxErrorMsg_r()`

```
GAIA GEO_DECLARE void gaiaSetGeosAuxErrorMsg_r (
    const void * p_cache,
    const char * msg )
```

Set the current GEOS (auxiliary) error message.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>msg</i>	the error message to be set.

See also

[gaiaSetAuxErrorMsg](#), [gaiaResetGeosMsg](#), [gaiaGetGeosErrorMsg](#), [gaiaGetGeosWarningMsg](#), [gaiaGetGeosAuxErrorMsg](#), [gaiaSetGeosWarningMsg](#), [gaiaSetGeosErrorMsg](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.188 `gaiaSetGeosErrorMsg()`

```
GAIA GEO_DECLARE void gaiaSetGeosErrorMsg (
    const char * msg )
```

Set the current GEOS error message.

Parameters

<i>msg</i>	the error message to be set.
------------	------------------------------

See also

[gaiaSetGeosErrorMsg_r](#), [gaiaResetGeosMsg](#), [gaiaGetGeosErrorMsg](#), [gaiaGetGeosWarningMsg](#), [gaiaGetGeosAuxErrorMsg](#), [gaiaSetGeosWarningMsg](#), [gaiaSetGeosAuxErrorMsg](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.189 gaiaSetGeosErrorMsg_r()

```
GAIA GEO_DECLARE void gaiaSetGeosErrorMsg_r (
    const void * p_cache,
    const char * msg )
```

Set the current GEOS error message.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>msg</i>	the error message to be set.

See also

[gaiaSetGeosErrorMsg](#), [gaiaResetGeosMsg](#), [gaiaGetGeosErrorMsg](#), [gaiaGetGeosWarningMsg](#), [gaiaGetGeosAuxErrorMsg](#), [gaiaSetGeosWarningMsg](#), [gaiaSetGeosAuxErrorMsg](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.190 gaiaSetGeosWarningMsg()

```
GAIA GEO_DECLARE void gaiaSetGeosWarningMsg (
    const char * msg )
```

Set the current GEOS warning message.

Parameters

<i>msg</i>	the warning message to be set.
------------	--------------------------------

See also

[gaiaSetGeosWarningMsg_r](#), [gaiaResetGeosMsg](#), [gaiaGetGeosErrorMsg](#), [gaiaGetGeosWarningMsg](#),
[gaiaGetGeosAuxErrorMsg](#), [gaiaSetGeosErrorMsg](#), [gaiaSetGeosAuxErrorMsg](#)

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.191 gaiaSetGeosWarningMsg_r()

```
GAIA GEO_DECLARE void gaiaSetGeosWarningMsg_r (
    const void * p_cache,
    const char * msg )
```

Set the current GEOS warning message.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>msg</i>	the warning message to be set.

See also

[gaiaSetGeosWarningMsg](#), [gaiaResetGeosMsg](#), [gaiaGetGeosErrorMsg](#), [gaiaGetGeosWarningMsg](#),
[gaiaGetGeosAuxErrorMsg](#), [gaiaSetGeosErrorMsg](#), [gaiaSetGeosAuxErrorMsg](#)

Note

reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.192 gaiaSetProjDatabasePath()

```
GAIA GEO_DECLARE const char* gaiaSetProjDatabasePath (
    const void * p_cache,
    const char * path )
```

Sets the PATH leading to the private PROJ.6 database.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>path</i>	the pathname to be set

Returns

the currently set PATH leading to the private PROJ.6 database; NULL if no PROJ database is defined.

See also

[gaiaGetProjDatabasePath](#)

Note

reentrant and thread-safe.

Remarks

PROJ.6 support required.

6.5.2.193 gaiaSetProjErrorMsg_r()

```
GAIGEO_DECLARE void gaiaSetProjErrorMsg_r (
    const void * p_cache,
    const char * msg )
```

Set the current PROJ error message.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>msg</i>	the error message to be set.

See also

[gaiaResetProjErrorMsg_r](#), [gaiaGetProjErrorMsg_r](#)

Note

reentrant and thread-safe.

Remarks

PROJ.6 support required.

6.5.2.194 `gaiaSetRtTopoErrorMsg()`

```
GAIA GEO_DECLARE void gaiaSetRtTopoErrorMsg (
    const void * p_cache,
    const char * msg )
```

Set the current RTTOPO error message.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>msg</i>	the error message to be set.

See also

[gaiaResetRtTopoMsg](#), [gaiaGetRtTopoErrorMsg](#), [gaiaGetRtTopoWarningMsg](#), [gaiaSetRtTopoWarningMsg](#)

Remarks

RTTOPO support required.

6.5.2.195 `gaiaSetRtTopoWarningMsg()`

```
GAIA GEO_DECLARE void gaiaSetRtTopoWarningMsg (
    const void * p_cache,
    const char * msg )
```

Set the current RTTOPO warning message.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>msg</i>	the warning message to be set.

See also

[gaiaResetRtTopoMsg](#), [gaiaGetRtTopoErrorMsg](#), [gaiaGetRtTopoWarningMsg](#), [gaiaSetRtTopoErrorMsg](#)

Remarks

RTTOPO support required.

6.5.2.196 gaiaSharedPaths()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSharedPaths (  
    gaiaGeomCollPtr geom1,  
    gaiaGeomCollPtr geom2 )
```

Spatial operator: Shared Paths.

Parameters

<i>geom1</i>	pointer to first Geometry object
<i>geom2</i>	pointer to second Geometry object

Returns

the pointer to newly created Geometry object representing any Share Path common to both input geometries: NULL on failure.

See also

[gaiaSharedPaths_r](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaSharedPaths\(\)](#) not reentrant and thread unsafe.

Remarks

GEOS-ADVANCED support required.

6.5.2.197 **gaiaSharedPaths_r()**

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSharedPaths\_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial operator: Shared Paths.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	pointer to first Geometry object
<i>geom2</i>	pointer to second Geometry object

Returns

the pointer to newly created Geometry object representing any Share Path common to both input geometries: NULL on failure.

See also

[gaiaSharedPaths](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaSharedPaths_r\(\)](#) reentrant and thread-safe.

Remarks

GEOS-ADVANCED support required.

6.5.2.198 gaiaShortestLine()

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaShortestLine (
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial operator: Shortest Line.

Parameters

<i>geom1</i>	pointer to the first Geometry object.
<i>geom2</i>	pointer to the second Geometry object.

Returns

the pointer to newly created Geometry object representing a Linestring; NULL on failure.
the returned Linestring graphically represents the minimum distance intercurring between both input geometries.

See also

[gaiaShortestLine_r](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaShortestLine\(\)](#) not reentrant and thread unsafe.

Remarks

GEOS-ADVANCED support required.

6.5.2.199 gaiaShortestLine_r()

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaShortestLine_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Spatial operator: Shortest Line.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	pointer to the first Geometry object.
<i>geom2</i>	pointer to the second Geometry object.

Returns

the pointer to newly created Geometry object representing a Linestring; NULL on failure.
the returned Linestring graphically represents the minimum distance intercurring between both input geometries.

See also

[gaiaShortestLine](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaShortestLine_r\(\)](#)
reentrant and thread-safe.

Remarks

GEOS-ADVANCED support required.

6.5.2.200 **gaiaSingleSidedBuffer()**

```
GAIA_GEO_DECLARE gaiaGeomCollPtr gaiaSingleSidedBuffer (
    gaiaGeomCollPtr geom,
    double radius,
    int points,
    int left_right )
```

Spatial operator: Single Sided Buffer.

Parameters

<i>geom</i>	the input Geometry object
<i>radius</i>	the buffer's radius
<i>points</i>	number of points (aka vertices) to be used in order to approximate a circular arc.
<i>left_right</i>	if set to 1 the left-sided Buffer will be returned; otherwise the right-sided one.

Returns

the pointer to newly created Geometry object representing the single-sided Buffer of input Geometry: NULL on failure.

See also

[gaiaSingleSidedBuffer_r](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaSingleSidedBuffer\(\)](#) not reentrant and thread unsafe.

Remarks

GEOS-ADVANCED support required.

6.5.2.201 [gaiaSingleSidedBuffer_r\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSingleSidedBuffer\_r (  
    const void * p_cache,  
    gaiaGeomCollPtr geom,  
    double radius,  
    int points,  
    int left_right )
```

Spatial operator: Single Sided Buffer.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	the input Geometry object
<i>radius</i>	the buffer's radius
<i>points</i>	number of points (aka vertices) to be used in order to approximate a circular arc.
<i>left_right</i>	if set to 1 the left-sided Buffer will be returned; otherwise the right-sided one.

Returns

the pointer to newly created Geometry object representing the single-sided Buffer of input Geometry: NULL on failure.

See also

[gaiaSingleSidedBuffer](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaSingleSidedBuffer_r\(\)](#) reentrant and thread-safe.

Remarks

GEOS-ADVANCED support required.

6.5.2.202 `gaiaSnap()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSnap (
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2,
    double tolerance )
```

Spatial operator: Snap.

Parameters

<i>geom1</i>	pointer to the first Geometry object.
<i>geom2</i>	pointer to the second Geometry object.
<i>tolerance</i>	approximation factor

Returns

the pointer to newly created Geometry object; NULL on failure.
the returned Geometry represents the first input Geometry (nicely *snapped* to the second input Geometry, whenever is possible).

See also

[gaiaSnap_r](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaSnap\(\)](#)
not reentrant and thread unsafe.

Remarks

GEOS-ADVANCED support required.

6.5.2.203 `gaiaSnap_r()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSnap_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2,
    double tolerance )
```

Spatial operator: Snap.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	pointer to the first Geometry object.
<i>geom2</i>	pointer to the second Geometry object.
<i>tolerance</i>	approximation factor

Returns

the pointer to newly created Geometry object; NULL on failure.
 the returned Geometry represents the first input Geometry (nicely *snapped* to the second input Geometry, whenever is possible).

See also

[gaiaSnap](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaSnap_r\(\)](#)
 reentrant and thread-safe.

Remarks

GEOS-ADVANCED support required.

6.5.2.204 gaiaSnapToGrid()

```
GAIA_GEO_DECLARE gaiaGeomCollPtr gaiaSnapToGrid (
    gaiaGeomCollPtr geom,
    double origin_x,
    double origin_y,
    double origin_z,
    double origin_m,
    double size_x,
    double size_y,
    double size_z,
    double size_m )
```

Utility function: SnapToGrid.

Parameters

<i>geom</i>	the input Geometry object.
<i>origin</i> ↔ <i>_x</i>	the X cordinate identifying the Grid Origin.
<i>origin</i> ↔ <i>_y</i>	the Y coordinate identifying the Grid Origin.
<i>origin</i> ↔ <i>_z</i>	the Z cordinate identifying the Grid Origin.
<i>origin</i> ↔ <i>_m</i>	the M coordinate identifying the Grid Origin.
<i>size_x</i>	Grid cell size (X axis).
<i>size_y</i>	Grid cell size (Y axis).
<i>size_z</i>	Grid cell size (Z axis).
<i>size_m</i>	Grid cell size (M axis).

Returns

the pointer to newly created Geometry object: NULL on failure.
 this function will return a modified geometry having all points snapped to a regular Grid defined by its origin and cell size.
 Consecutive points falling on the same cell will be removed, eventually returning NULL if output points are not enough to define a geometry of the given type.
 Collapsed geometries in a collection are stripped from it.
 Specify 0 as size for any dimension you don't want to snap to a grid.
 NULL will be returned if the passed argument is invalid.

See also

[gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaSnapToGrid\(\)](#)

6.5.2.205 gaiaSplit()

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaSplit (
    const void * p_cache,
    gaiaGeomCollPtr input,
    gaiaGeomCollPtr blade )
```

Utility function: Split.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>input</i>	the input Geometry object.
<i>blade</i>	the blade Geometry object.

Returns

the pointer to newly created Geometry object: NULL on failure.
 The function supports splitting a line by point, a line by line, a polygon by line.

See also

[gaiaFreeGeomColl](#), [gaiaSplitLeft](#), [gaiaSplitRight](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaSplit\(\)](#)
 gaiaSplit will return both the **left** and the **right** split halves at the same time.

Remarks

RTTOPO support required.

6.5.2.206 `gaiaSplitLeft()`

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaSplitLeft (
    const void * p_cache,
    gaiaGeomCollPtr input,
    gaiaGeomCollPtr blade )
```

Utility function: SplitLeft.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>input</i>	the input Geometry object.
<i>blade</i>	the blade Geometry object.

Returns

the pointer to newly created Geometry object: NULL on failure.
The function supports splitting a line by point, a line by line, a polygon by line.

See also

[gaiaFreeGeomColl](#), [gaiaSplit](#), [gaiaSplitRight](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaSplitLeft\(\)](#)
`gaiaSplitLeft` will only return the **left** split half; NULL may be eventually returned if empty.

Remarks

RTTOPO support required.

6.5.2.207 `gaiaSplitRight()`

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaSplitRight (
    const void * p_cache,
    gaiaGeomCollPtr input,
    gaiaGeomCollPtr blade )
```

Utility function: SplitRight.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>input</i>	the input Geometry object.
<i>blade</i>	the blade Geometry object.

Returns

the pointer to newly created Geometry object: NULL on failure.
 The function supports splitting a line by point, a line by line, a polygon by line.

See also

[gaiaFreeGeomColl](#), [gaiaSplit](#), [gaiaSplitLeft](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaSplitRight\(\)](#)

[gaiaSplitLeft](#) will only return the **right** split half; NULL may be eventually returned if empty.

Remarks

RTTOPO support required.

6.5.2.208 [gaiaSquareGrid\(\)](#)

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaSquareGrid (
    gaiaGeomCollPtr geom,
    double origin_x,
    double origin_y,
    double size,
    int mode )
```

Utility function: [SquareGrid](#).

Parameters

<i>geom</i>	the Geometry to be covered by the Grid.
<i>origin_x</i>	the X coordinate identifying the Grid Origin.
<i>origin_y</i>	the Y coordinate identifying the Grid Origin.
<i>size</i>	the Grid cell-side size.
<i>mode</i>	any positive value will return a MULTILINESTRING, any will return a MULTIPOINT; zero will return a MULTIPOLYGON containing square POLYGONS.

Returns

the pointer to newly created Geometry object: NULL on failure.
 this function will always return a MultiPolygon
 NULL will be returned if any argument is invalid.

See also

[gaiaSquareGrid_r](#), [gaiaFreeGeomColl](#), [gaiaTriangularGrid](#), [gaiaHexagonalGrid](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaSquareGrid\(\)](#)
 not reentrant and thread unsafe.

6.5.2.209 gaiaSquareGrid_r()

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaSquareGrid_r (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    double origin_x,
    double origin_y,
    double size,
    int mode )
```

Utility function: SquareGrid.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	the Geometry to be covered by the Grid.
<i>origin_x</i>	the X coordinate identifying the Grid Origin.
<i>origin_y</i>	the Y coordinate identifying the Grid Origin.
<i>size</i>	the Grid cell-side size.
<i>mode</i>	any positive value will return a MULTILINESTRING, any will return a MULTIPOINT; zero will return a MULTIPOLYGON containing square POLYGONS.

Returns

the pointer to newly created Geometry object: NULL on failure.
 this function will always return a MultiPolygon
 NULL will be returned if any argument is invalid.

See also

[gaiaSquareGrid](#), [gaiaFreeGeomColl](#), [gaiaTriangularGrid](#), [gaiaHexagonalGrid](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaSquareGrid_r\(\)](#) reentrant and thread-safe.

6.5.2.210 gaiaSubdivide()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSubdivide (
    const void * p_cache,
    gaiaGeomCollPtr input,
    int max_vertices )
```

Utility function: subdividing Geometry.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>input</i>	the input Geometry object.
<i>max_vertices</i>	the maximum number of vertices for each part in the output geometry that will be returned.

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaSubdivide\(\)](#)

Remarks

RTTOPO support required.

6.5.2.211 gaiaToGeos()

```
GAIAGEO_DECLARE void* gaiaToGeos (
    const gaiaGeomCollPtr gaia )
```

Converts a Geometry object into a GEOS Geometry.

Parameters

<i>gaia</i>	pointer to Geometry object
-------------	----------------------------

Returns

handle to GEOS Geometry

See also

[gaiaToGeos_r](#), [gaiaFromGeos_XY](#), [gaiaFromGeos_XYZ](#), [gaiaFromGeos_XYM](#), [gaiaFromGeos_XYZM](#), [gaiaToGeosSelective](#)

Note

convenience method, simply defaulting to `gaiaToGeosSelective(geom, GAIA2GEOS_ALL)`
not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.212 `gaiaToGeos_r()`

```
GAIA GEO_DECLARE void* gaiaToGeos_r (
    const void * p_cache,
    const gaiaGeomCollPtr gaia )
```

Converts a Geometry object into a GEOS Geometry.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>gaia</i>	pointer to Geometry object

Returns

handle to GEOS Geometry

See also

[gaiaToGeos](#), [gaiaFromGeos_XY](#), [gaiaFromGeos_XYZ](#), [gaiaFromGeos_XYM](#), [gaiaFromGeos_XYZM](#), [gaiaToGeosSelective_r](#)

Note

convenience method, simply defaulting to `gaiaToGeosSelective_r(p_cache, geom, GAIA2GEOS_ALL)` reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.213 gaiaToGeosSelective()

```
GAIA2GEO_DECLARE void* gaiaToGeosSelective (
    const gaiaGeomCollPtr gaia,
    int mode )
```

Converts a Geometry object into a GEOS Geometry.

Parameters

<i>gaia</i>	pointer to Geometry object
<i>mode</i>	one of GAIA2GEOS_ALL, GAIA2GEOS_ONLY_POINTS, GAIA2GEOS_ONLY_LINESTRINGS or GAIA2GEOS_ONLY_POLYGONS

Returns

handle to GEOS Geometry

See also

[gaiaToGeosSelective_r](#), [gaiaFromGeos_XY](#), [gaiaFromGeos_XYZ](#), [gaiaFromGeos_XYM](#), [gaiaFromGeos_XYZM](#)

Note

if the mode argument is not GAIA2GEOS_ALL only elementary geometries of the selected type will be passed to GEOS, ignoring any other.
not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.214 gaiaToGeosSelective_r()

```
GAIA2GEO_DECLARE void* gaiaToGeosSelective_r (
    const void * p_cache,
    const gaiaGeomCollPtr gaia,
    int mode )
```

Converts a Geometry object into a GEOS Geometry.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>gaia</i>	pointer to Geometry object
<i>mode</i>	one of GAIA2GEOS_ALL, GAIA2GEOS_ONLY_POINTS, GAIA2GEOS_ONLY_LINESTRINGS or GAIA2GEOS_ONLY_POLYGONS

Returns

handle to GEOS Geometry

See also

[gaiaToGeosSelective](#), [gaiaFromGeos_XY](#), [gaiaFromGeos_XYZ](#), [gaiaFromGeos_XYM](#), [gaiaFromGeos_XYZM](#)

Note

if the mode argument is not GAIA2GEOS_ALL only elementary geometries of the selected type will be passed to GEOS, ignoring any other.
reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.215 **gaiaToTWKB()**

```

GAIAGEO_DECLARE int gaiaToTWKB (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    unsigned char precision_xy,
    unsigned char precision_z,
    unsigned char precision_m,
    int with_size,
    int with_bbox,
    unsigned char ** twkb,
    int * size_twkb )

```

Converts a native binary Geometry into a compressed TWKB Geometry.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	the Geometry to be converted
<i>precision_xy</i>	how much precision (decimal digits) X and Y
<i>precision_z</i>	how much precision (decimal digits) Z
<i>precision_m</i>	how much precision (decimal digits) M
<i>with_size</i>	including sizes into the TWKB
<i>with_bbox</i>	including a BBOX into the TWKB
<i>twkb</i>	on succesfull completion this pointer will reference the TWKB geometry
<i>size_twkb</i>	on succesfull completion this pointer will reference the size (in bytes) of the TWKB geometry

Returns

0 on failure: any other value on success.

See also

[gaiaFromTWKB](#)

Note

you are responsible to free (before or after) the TWKB geometry created by [gaiaToTWKB\(\)](#).

Remarks

RTTOPO support required.

6.5.2.216 gaiaTransform()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaTransform (
    gaiaGeomCollPtr org,
    const char * proj_from,
    const char * proj_to )
```

Transforms a Geometry object into a different Reference System [aka Reprojection].

Parameters

<i>org</i>	pointer to input Geometry object.
<i>proj_from</i>	geodetic parameters string [PROJ.4 format] qualifying the input Reference System
<i>proj_to</i>	geodetic parameters string [PROJ.4 format] qualifying the output Reference System

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaTransform_r](#), [gaiaTransformXY](#), [gaiaTransformXYZ](#), [gaiaTransformEx](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry,
this including any Geometry returned by [gaiaGeometryTransform\(\)](#)
not reentrant and thread unsafe.

Remarks

PROJ support required

6.5.2.217 gaiaTransform_r()

```

GAIA GEO_DECLARE gaiaGeomCollPtr gaiaTransform_r (
    const void * p_cache,
    gaiaGeomCollPtr org,
    const char * proj_from,
    const char * proj_to )

```

Transforms a Geometry object into a different Reference System [aka Reprojection].

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>org</i>	pointer to input Geometry object.
<i>proj_from</i>	geodetic parameters string [PROJ.4 format] qualifying the input Reference System
<i>proj_to</i>	geodetic parameters string [PROJ.4 format] qualifying the output Reference System

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaTransform](#), [gaiaTransformXY_r](#), [gaiaTransformXYZ_r](#), [gaiaTransformEx_r](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaGeometryTransform\(\)](#) reentrant and thread-safe.

Remarks

PROJ support required

6.5.2.218 gaiaTransformEx()

```

GAIA GEO_DECLARE gaiaGeomCollPtr gaiaTransformEx (
    gaiaGeomCollPtr org,
    const char * proj_string_1,
    const char * proj_string_2,
    gaiaProjAreaPtr proj_bbox )

```

Transforms a Geometry object into a different Reference System [aka Reprojection].

Parameters

<i>org</i>	pointer to input Geometry object.
<i>proj_string_1</i>	any valid string accepted by PROJ.6 for identifying the origin CRS of a transformation, or a string defining a transformation pipeline.
<i>proj_string_2</i>	any valid string accepted by PROJ.6 for identifying the destination CRS of a transformation. expected to be NULL if <i>proj_string_1</i> defines a pipeline.
<i>proj_bbox</i>	pointer to a BoundingBox object defining the specific "area of use" of the transformation.

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaTransform](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by `gaiaGeometryTransformEx()` not reentrant and thread unsafe.

Remarks

PROJ.6 support required

6.5.2.219 gaiaTransformEx_r()

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaTransformEx_r (
    const void * p_cache,
    gaiaGeomCollPtr org,
    const char * proj_string_1,
    const char * proj_string_2,
    gaiaProjAreaPtr proj_bbox )
```

Transforms a Geometry object into a different Reference System [aka Reprojection].

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>org</i>	pointer to input Geometry object.
<i>proj_string_1</i>	any valid string accepted by PROJ.6 for identifying the origin CRS of a transformation, or a string defining a transformation pipeline. can never be NULL.
<i>proj_string_2</i>	any valid string accepted by PROJ.6 for identifying the destination CRS of a transformation. expected to be NULL if <i>proj_string_1</i> defines a pipeline.
<i>proj_bbox</i>	pointer to a BoundingBox object defining the specific "area of use" of the transformation. may be NULL.

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaTransform_r](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by `gaiaGeometryTransformEx()` reentrant and thread-safe.

Remarks

PROJ.6 support required

6.5.2.220 gaiaTransformXY()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaTransformXY (
    gaiaGeomCollPtr org,
    const char * proj_from,
    const char * proj_to )
```

Transforms a Geometry object into a different Reference System [aka Reprojection] This is a special "flavor" of [gaiaTransform\(\)](#) just considering X and Y coordinates; Z and M values will be left untouched.

Mainly intended as a workaround possibly useful when facing partially broken PROJ definitions.

Parameters

<i>org</i>	pointer to input Geometry object.
<i>proj_from</i>	geodetic parameters string [EPSG format] qualifying the input Reference System
<i>proj_to</i>	geodetic parameters string [EPSG format] qualifying the output Reference System

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaTransformXY_r](#), [gaiaTransformXYZ_r](#), [gaiaTransform](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by `gaiaGeometryTransform()` not reentrant and thread unsafe.

Remarks

PROJ support required

6.5.2.221 `gaiaTransformXY_r()`

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaTransformXY_r (
    const void * p_cache,
    gaiaGeomCollPtr org,
    const char * proj_from,
    const char * proj_to )
```

Transforms a Geometry object into a different Reference System [aka Reprojection] This is a special "flavor" of [gaiaTransform_r\(\)](#) just considering X and Y coordinates; Z and M values will be left untouched.

Mainly intended as a workaround possibly useful when facing partially broken PROJ definitions.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>org</i>	pointer to input Geometry object.
<i>proj_from</i>	geodetic parameters string [EPSG format] qualifying the input Reference System
<i>proj_to</i>	geodetic parameters string [EPSG format] qualifying the output Reference System

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaTransformXY](#), [gaiaTransformXYZ](#), [gaiaTransform_r](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by `gaiaGeometryTransform()`
reentrant and thread-safe.

Remarks

PROJ support required

6.5.2.222 `gaiaTransformXYZ()`

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaTransformXYZ (
    gaiaGeomCollPtr org,
    const char * proj_from,
    const char * proj_to )
```

Transforms a Geometry object into a different Reference System [aka Reprojection] This is a special "flavor" of [gaiaTransform\(\)](#) just considering X, Y and Z coordinates; M values will be left untouched.

Mainly intended as a workaround possibly useful when handling 4D geometries having M-values not corresponding to Time.

Parameters

<i>org</i>	pointer to input Geometry object.
<i>proj_from</i>	geodetic parameters string [EPSG format] qualifying the input Reference System
<i>proj_to</i>	geodetic parameters string [EPSG format] qualifying the output Reference System

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaTransformXY_r](#), [gaiaTransformXYZ_r](#), [gaiaTransform](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaGeometryTransform\(\)](#)
not reentrant and thread unsafe.

Remarks

PROJ support required

6.5.2.223 [gaiaTransformXYZ_r\(\)](#)

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaTransformXYZ\_r (
    const void * p_cache,
    gaiaGeomCollPtr org,
    const char * proj_from,
    const char * proj_to )
```

Transforms a Geometry object into a different Reference System [aka Reprojection] This is a special "flavor" of [gaiaTransform\(\)](#) just considering X, Y and Z coordinates; M values will be left untouched.

Mainly intended as a workaround possibly useful when handling 4D geometries having M-values not corresponding to Time.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>org</i>	pointer to input Geometry object.
<i>proj_from</i>	geodetic parameters string [EPSG format] qualifying the input Reference System
<i>proj_to</i>	geodetic parameters string [EPSG format] qualifying the output Reference System

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaTransformXY](#), [gaiaTransformXYZ](#), [gaiaTransform_r](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by `gaiaGeometryTransform()`
reentrant and thread-safe.

Remarks

PROJ support required

6.5.2.224 gaiaTriangularGrid()

```
GAIA_GEO_DECLARE gaiaGeomCollPtr gaiaTriangularGrid (
    gaiaGeomCollPtr geom,
    double origin_x,
    double origin_y,
    double size,
    int mode )
```

Utility function: TriangularGrid.

Parameters

<i>geom</i>	the Geometry to be covered by the Grid.
<i>origin</i> ↔ <i>_x</i>	the X coordinate identifying the Grid Origin.
<i>origin</i> ↔ <i>_y</i>	the Y coordinate identifying the Grid Origin.
<i>size</i>	the Grid cell-side size.
<i>mode</i>	any positive value will return a MULTILINESTRING, any will return a MULTIPOINT; zero will return a MULTIPOLYGON containing triangular POLYGONS.

Returns

the pointer to newly created Geometry object: NULL on failure.
this function will always return a MultiPolygon
NULL will be returned if any argument is invalid.

See also

[gaiaTriangularGrid_r](#), [gaiaFreeGeomColl](#), [gaiaSquareGrid](#), [gaiaHexagonalGrid](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaTriangularGrid\(\)](#)
not reentrant and thread unsafe.

6.5.2.225 gaiaTriangularGrid_r()

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaTriangularGrid_r (
    const void * p_cache,
    gaiaGeomCollPtr geom,
    double origin_x,
    double origin_y,
    double size,
    int mode )
```

Utility function: TriangularGrid.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	the Geometry to be covered by the Grid.
<i>origin_x</i>	the X coordinate identifying the Grid Origin.
<i>origin_y</i>	the Y coordinate identifying the Grid Origin.
<i>size</i>	the Grid cell-side size.
<i>mode</i>	any positive value will return a MULTILINESTRING, any will return a MULTIPOINT; zero will return a MULTIPOLYGON containing triangular POLYGONS.

Returns

the pointer to newly created Geometry object: NULL on failure.
this function will always return a MultiPolygon
NULL will be returned if any argument is invalid.

See also

[gaiaTriangularGrid](#), [gaiaFreeGeomColl](#), [gaiaSquareGrid](#), [gaiaHexagonalGrid](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaTriangularGrid_r\(\)](#)
reentrant and thread-safe.

6.5.2.226 gaiaUnaryUnion()

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaUnaryUnion (
    gaiaGeomCollPtr geom )
```

Spatial operator: Unary Union.

Parameters

<i>geom</i>	the input Geometry object.
-------------	----------------------------

Returns

the pointer to newly created Geometry object: NULL on failure.
 this function is the same as `gaiaGeometryUnion`, except in that this works internally to the input Geometry itself. NULL on failure.

See also

[gaiaUnaryUnion_r](#), [gaiaFreeGeomColl](#), [gaiaGeometryUnion](#), [gaiaUnionCascaded](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaUnaryUnion\(\)](#)
 not reentrant and thread unsafe.

Remarks

GEOS-ADVANCED support required.

6.5.2.227 `gaiaUnaryUnion_r()`

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaUnaryUnion_r (
    const void * p_cache,
    gaiaGeomCollPtr geom )
```

Spatial operator: Unary Union.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	the input Geometry object.

Returns

the pointer to newly created Geometry object: NULL on failure.
 this function is the same as `gaiaGeometryUnion`, except in that this works internally to the input Geometry itself. NULL on failure.

See also

[gaiaUnaryUnion](#), [gaiaFreeGeomColl](#), [gaiaGeometryUnion](#), [gaiaUnionCascaded](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaUnaryUnion_r\(\)](#) reentrant and thread-safe.

Remarks

GEOS-ADVANCED support required.

6.5.2.228 [gaiaUnionCascaded\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaUnionCascaded (  
    gaiaGeomCollPtr geom )
```

Spatial operator: Union Cascaded.

Parameters

<i>geom</i>	the input Geometry object.
-------------	----------------------------

Returns

the pointer to newly created Geometry object: NULL on failure.
this function is similar to [gaiaUnaryUnion](#), but it only accepts Polygons and MultiPolygons and it's now deprecated; anyway it's supported on older GEOS versions. NULL on failure.

See also

[gaiaUnionCascaded](#), [gaiaFreeGeomColl](#), [gaiaGeometryUnion](#), [gaiaUnionUnion](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaUnionCascaded\(\)](#) not reentrant and thread unsafe.

Remarks

GEOS support required.

6.5.2.229 [gaiaUnionCascaded_r\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaUnionCascaded\_r (  
    const void * p_cache,  
    gaiaGeomCollPtr geom )
```

Spatial operator: Union Cascaded.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	the input Geometry object.

Returns

the pointer to newly created Geometry object: NULL on failure.
 this function is similar to [gaiaUnaryUnion](#), but it only accepts Polygons and MultiPolygons and it's now deprecated; anyway it's supported on older GEOS versions. NULL on failure.

See also

[gaiaUnionCascaded](#), [gaiaFreeGeomColl](#), [gaiaGeometryUnion](#), [gaiaUnionUnion](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaUnionCascaded_r\(\)](#)
 reentrant and thread-safe.

Remarks

GEOS support required.

6.5.2.230 [gaiaVoronoiDiagram\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaVoronoiDiagram (
    gaiaGeomCollPtr geom,
    double extra_frame_size,
    double tolerance,
    int only_edges )
```

Voronoi Diagram.

Parameters

<i>geom</i>	pointer to input Geometry object.
<i>extra_frame_size</i>	percent factor expanding the BBOX of input Geometry
<i>tolerance</i>	optional snapping tolerance.
<i>only_edges</i>	if non-zero will return a MULTILINESTRING, otherwise it will return a MULTIPOLYGON.

Returns

the pointer to newly created Geometry object: NULL on failure.
 NULL will be returned if any argument is invalid.

See also

[gaiaVoronoiDiagram_r](#), [gaiaFreeGeomColl](#), [gaiaDelaunayTriangulation](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaVoronoiDiagram\(\)](#) not reentrant and thread unsafe.

Remarks

GEOS-TRUNK support required.

6.5.2.231 [gaiaVoronoiDiagram_r\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaVoronoiDiagram\_r (  
    const void * p_cache,  
    gaiaGeomCollPtr geom,  
    double extra_frame_size,  
    double tolerance,  
    int only_edges )
```

Voronoi Diagram.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to input Geometry object.
<i>extra_frame_size</i>	percent factor expanding the BBOX of input Geometry
<i>tolerance</i>	optional snapping tolerance.
<i>only_edges</i>	if non-zero will return a MULTILINESTRING, otherwise it will return a MULTIPOLYGON.

Returns

the pointer to newly created Geometry object: NULL on failure.
NULL will be returned if any argument is invalid.

See also

[gaiaVoronoiDiagram](#), [gaiaFreeGeomColl](#), [gaiaDelaunayTriangulation](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaVoronoiDiagram_r\(\)](#) reentrant and thread-safe.

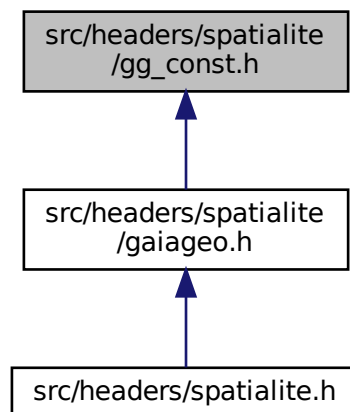
Remarks

GEOS-TRUNK support required.

6.6 src/headers/spatialite/gg_const.h File Reference

Geometry constants and macros.

This graph shows which files directly or indirectly include this file:



Macros

- `#define GAIA_VECTORS_LIST_FAST 0`
mode: FAST (QGIS data-provider)
- `#define GAIA_VECTORS_LIST_OPTIMISTIC 1`
mode: OPTIMISTIC
- `#define GAIA_VECTORS_LIST_PESSIMISTIC 2`
mode: PESSIMISTIC
- `#define GAIA_VECTOR_UNKNOWN -1`
Vector Layer: unknown type.
- `#define GAIA_VECTOR_TABLE 1`
Vector Layer: Spatial Table.
- `#define GAIA_VECTOR_VIEW 2`
Vector Layer: Spatial View.
- `#define GAIA_VECTOR_VIRTUAL 3`
Vector Layer: Virtual Shape.
- `#define GAIA_VECTOR_GEOMETRY 0`
Vector Layer Geometry: Geometry.
- `#define GAIA_VECTOR_POINT 1`
Vector Layer Geometry: Point.
- `#define GAIA_VECTOR_LINESTRING 2`
Vector Layer Geometry: Linestring.
- `#define GAIA_VECTOR_POLYGON 3`
Vector Layer Geometry: Polygon.
- `#define GAIA_VECTOR_MULTIPPOINT 4`

- *Vector Layer Geometry: MultiPoint.*
- #define [GAIA_VECTOR_MULTILINESTRING](#) 5
- *Vector Layer Geometry: MultiLinestring.*
- #define [GAIA_VECTOR_MULTIPOLYGON](#) 6
- *Vector Layer Geometry: MultiPolygon.*
- #define [GAIA_VECTOR_GEOMETRYCOLLECTION](#) 7
- *Vector Layer Geometry: GeometryCollection.*
- #define [GAIA_SPATIAL_INDEX_NONE](#) 0
- *Vector Layer: no Spatial Index.*
- #define [GAIA_SPATIAL_INDEX_RTREE](#) 1
- *Vector Layer: Spatial Index RTree.*
- #define [GAIA_SPATIAL_INDEX_MBRCACHE](#) 2
- *Vector Layer: Spatial Index MbrCache.*
- #define [GAIA_TYPE_NONE](#) 0
- *WKT parser: unknown Geometry type.*
- #define [GAIA_TYPE_POINT](#) 1
- *WKT parser: Point Geometry type.*
- #define [GAIA_TYPE_LINESTRING](#) 2
- *WKT parser: Linestring Geometry type.*
- #define [GAIA_TYPE_POLYGON](#) 3
- *WKT parser: Polygon Geometry type.*
- #define [GAIA_BIG_ENDIAN](#) 0
- *Big-Endian marker.*
- #define [GAIA_LITTLE_ENDIAN](#) 1
- *Little-Endian marker.*
- #define [GAIA_TINYPOINT_BIG_ENDIAN](#) 0x80
- *TinyPoint Big-Endian marker.*
- #define [GAIA_TINYPOINT_LITTLE_ENDIAN](#) 0x81
- *TinyPoint Little-Endian marker.*
- #define [GAIA_MARK_START](#) 0x00
- *BLOB-Geometry internal marker: START.*
- #define [GAIA_MARK_END](#) 0xFE
- *BLOB-Geometry internal marker: END.*
- #define [GAIA_MARK_MBR](#) 0x7C
- *BLOB-Geometry internal marker: MBR.*
- #define [GAIA_MARK_ENTITY](#) 0x69
- *BLOB-Geometry internal marker: ENTITY.*
- #define [GAIA_UNKNOWN](#) 0
- *BLOB-Geometry CLASS: unknown.*
- #define [GAIA_POINT](#) 1
- *BLOB-Geometry CLASS: POINT.*
- #define [GAIA_LINESTRING](#) 2
- *BLOB-Geometry CLASS: LINESTRING.*
- #define [GAIA_POLYGON](#) 3
- *BLOB-Geometry CLASS: POLYGON.*
- #define [GAIA_MULTIPOINT](#) 4
- *BLOB-Geometry CLASS: MULTIPOINT.*
- #define [GAIA_MULTILINESTRING](#) 5
- *BLOB-Geometry CLASS: MULTILINESTRING.*
- #define [GAIA_MULTIPOLYGON](#) 6
- *BLOB-Geometry CLASS: MULTIPOLYGON.*

- #define [GAIA_GEOMETRYCOLLECTION](#) 7
BLOB-Geometry CLASS: GEOMETRYCOLLECTION.
- #define [GAIA_POINTZ](#) 1001
BLOB-Geometry CLASS: POINT Z.
- #define [GAIA_LINESTRINGZ](#) 1002
BLOB-Geometry CLASS: LINESTRING Z.
- #define [GAIA_POLYGONZ](#) 1003
BLOB-Geometry CLASS: POLYGON Z.
- #define [GAIA_MULTIPPOINTZ](#) 1004
BLOB-Geometry CLASS: MULTIPPOINT Z.
- #define [GAIA_MULTILINESTRINGZ](#) 1005
BLOB-Geometry CLASS: MULTILINESTRING Z.
- #define [GAIA_MULTIPOLYGONZ](#) 1006
BLOB-Geometry CLASS: MULTIPOLYGON Z.
- #define [GAIA_GEOMETRYCOLLECTIONZ](#) 1007
BLOB-Geometry CLASS: GEOMETRYCOLLECTION Z.
- #define [GAIA_POINTM](#) 2001
BLOB-Geometry CLASS: POINT M.
- #define [GAIA_LINESTRINGM](#) 2002
BLOB-Geometry CLASS: LINESTRING M.
- #define [GAIA_POLYGONM](#) 2003
BLOB-Geometry CLASS: POLYGON M.
- #define [GAIA_MULTIPPOINTM](#) 2004
BLOB-Geometry CLASS: MULTIPPOINT M.
- #define [GAIA_MULTILINESTRINGM](#) 2005
BLOB-Geometry CLASS: MULTILINESTRING M.
- #define [GAIA_MULTIPOLYGONM](#) 2006
BLOB-Geometry CLASS: MULTIPOLYGON M.
- #define [GAIA_GEOMETRYCOLLECTIONM](#) 2007
BLOB-Geometry CLASS: GEOMETRYCOLLECTION M.
- #define [GAIA_POINTZM](#) 3001
BLOB-Geometry CLASS: POINT ZM.
- #define [GAIA_LINESTRINGZM](#) 3002
BLOB-Geometry CLASS: LINESTRING ZM.
- #define [GAIA_POLYGONZM](#) 3003
BLOB-Geometry CLASS: POLYGON ZM.
- #define [GAIA_MULTIPPOINTZM](#) 3004
BLOB-Geometry CLASS: MULTIPPOINT ZM.
- #define [GAIA_MULTILINESTRINGZM](#) 3005
BLOB-Geometry CLASS: MULTILINESTRING ZM.
- #define [GAIA_MULTIPOLYGONZM](#) 3006
BLOB-Geometry CLASS: MULTIPOLYGON ZM.
- #define [GAIA_GEOMETRYCOLLECTIONZM](#) 3007
BLOB-Geometry CLASS: GEOMETRYCOLLECTION ZM.
- #define [GAIA_TINYPOINT_XY](#) 0x01
BLOB-TinyPoint Type: POINT XY.
- #define [GAIA_TINYPOINT_XYZ](#) 0x02
BLOB-TinyPoint Type: POINT XYZ.
- #define [GAIA_TINYPOINT_XYM](#) 0x03
BLOB-TinyPoint Type: POINT XYM.
- #define [GAIA_TINYPOINT_XYZM](#) 0x04

- *BLOB-TinyPoint Type: POINT XYZM.*
- #define [GAIA_COMPRESSED_LINESTRING](#) 1000002
BLOB-Geometry CLASS: compressed LINESTRING.
- #define [GAIA_COMPRESSED_POLYGON](#) 1000003
BLOB-Geometry CLASS: compressed POLYGON.
- #define [GAIA_COMPRESSED_LINESTRINGZ](#) 1001002
BLOB-Geometry CLASS: compressed LINESTRING Z.
- #define [GAIA_COMPRESSED_POLYGONZ](#) 1001003
BLOB-Geometry CLASS: compressed POLYGON Z.
- #define [GAIA_COMPRESSED_LINESTRINGM](#) 1002002
BLOB-Geometry CLASS: compressed LINESTRING M.
- #define [GAIA_COMPRESSED_POLYGONM](#) 1002003
BLOB-Geometry CLASS: compressed POLYGON M.
- #define [GAIA_COMPRESSED_LINESTRINGZM](#) 1003002
BLOB-Geometry CLASS: compressed LINESTRING ZM.
- #define [GAIA_COMPRESSED_POLYGONZM](#) 1003003
BLOB-Geometry CLASS: compressed POLYGON ZM.
- #define [GAIA_GEOSWKB_POINTZ](#) -2147483647
GEOS-WKB 3D CLASS: POINT Z.
- #define [GAIA_GEOSWKB_LINESTRINGZ](#) -2147483646
GEOS-WKB 3D CLASS: LINESTRING Z.
- #define [GAIA_GEOSWKB_POLYGONZ](#) -2147483645
GEOS-WKB 3D CLASS: POLYGON Z.
- #define [GAIA_GEOSWKB_MULTIPPOINTZ](#) -2147483644
GEOS-WKB 3D CLASS: MULTIPOINT Z.
- #define [GAIA_GEOSWKB_MULTILINESTRINGZ](#) -2147483643
GEOS-WKB 3D CLASS: MULTILINESTRING Z.
- #define [GAIA_GEOSWKB_MULTIPOLYGONZ](#) -2147483642
GEOS-WKB 3D CLASS: MULTIPOLYGON Z.
- #define [GAIA_GEOSWKB_GEOMETRYCOLLECTIONZ](#) -2147483641
GEOS-WKB 3D CLASS: POINT Z.
- #define [GAIA_NULL_VALUE](#) 0
DBF data type: NULL.
- #define [GAIA_TEXT_VALUE](#) 1
DBF data type: TEXT.
- #define [GAIA_INT_VALUE](#) 2
DBF data type: INT.
- #define [GAIA_DOUBLE_VALUE](#) 3
DBF data type: DOUBLE.
- #define [GAIA_START_POINT](#) 1
Linestring/Ring functions: START POINT.
- #define [GAIA_END_POINT](#) 2
Linestring/Ring functions: END POINT.
- #define [GAIA_POINTN](#) 3
Linestring/Ring functions: POINTN.
- #define [GAIA_MBR_CONTAINS](#) 1
MBR relationships: CONTAINS.
- #define [GAIA_MBR_DISJOINT](#) 2
MBR relationships: DISJOINT.
- #define [GAIA_MBR_EQUAL](#) 3
MBR relationships: EQUAL.

- #define [GAIA_MBR_INTERSECTS](#) 4
MBR relationships: INTERSECTS.
- #define [GAIA_MBR_OVERLAPS](#) 5
MBR relationships: OVERLAP.
- #define [GAIA_MBR_TOUCHES](#) 6
MBR relationships: TOUCHES.
- #define [GAIA_MBR_WITHIN](#) 7
MBR relationships: WITHIN.
- #define [GAIA_FILTER_MBR_WITHIN](#) 74
FilerMBR relationships: WITHIN.
- #define [GAIA_FILTER_MBR_CONTAINS](#) 77
FilerMBR relationships: CONTAINS.
- #define [GAIA_FILTER_MBR_INTERSECTS](#) 79
FilerMBR relationships: INTERSECTS.
- #define [GAIA_FILTER_MBR_DECLARE](#) 89
FilerMBR relationships: DECLARE.
- #define [GAIA_SVG_DEFAULT_RELATIVE](#) 0
SVG precision: RELATIVE.
- #define [GAIA_SVG_DEFAULT_PRECISION](#) 6
SVG precision: DEFAULT.
- #define [GAIA_SVG_DEFAULT_MAX_PRECISION](#) 15
SVG precision: MAX.
- #define [GAIA_NET_START](#) 0x67
VirtualNetwork internal markers: START.
- #define [GAIA_NET64_START](#) 0x68
VirtualNetwork internal markers: 64 bit START.
- #define [GAIA_NET64_A_STAR_START](#) 0x69
VirtualNetwork internal markers: A-Stat START.
- #define [GAIA_NET_END](#) 0x87
VirtualNetwork internal markers: END.
- #define [GAIA_NET_HEADER](#) 0xc0
VirtualNetwork internal markers: HEADER.
- #define [GAIA_NET_CODE](#) 0xa6
VirtualNetwork internal markers: CODE.
- #define [GAIA_NET_ID](#) 0xb5
VirtualNetwork internal markers: ID.
- #define [GAIA_NET_NODE](#) 0xde
VirtualNetwork internal markers: NODE.
- #define [GAIA_NET_ARC](#) 0x54
VirtualNetwork internal markers: ARC.
- #define [GAIA_NET_TABLE](#) 0xa0
VirtualNetwork internal markers: TABLE.
- #define [GAIA_NET_FROM](#) 0xa1
VirtualNetwork internal markers: FROM.
- #define [GAIA_NET_TO](#) 0xa2
VirtualNetwork internal markers: TO.
- #define [GAIA_NET_GEOM](#) 0xa3
VirtualNetwork internal markers: GEOM.
- #define [GAIA_NET_NAME](#) 0xa4
VirtualNetwork internal markers: NAME.
- #define [GAIA_NET_A_STAR_COEFF](#) 0xa5

- VirtualNetwork internal markers: COEFF.*
 - #define [GAIA_NET_BLOCK](#) 0xed
 - VirtualNetwork internal markers: BLOCK.*
 - #define [GAIA_XY](#) 0x00
 - Coordinate Dimensions: XY.*
 - #define [GAIA_XY_Z](#) 0x01
 - Coordinate Dimensions: XYZ.*
 - #define [GAIA_XY_M](#) 0x02
 - Coordinate Dimensions: XYM.*
 - #define [GAIA_XY_Z_M](#) 0x03
 - Coordinate Dimensions: XYZM.*
 - #define [GAIA_KM](#) 0
 - Length unit conversion: Kilometer.*
 - #define [GAIA_M](#) 1
 - Length unit conversion: Meter.*
 - #define [GAIA_DM](#) 2
 - Length unit conversion: Decimeter.*
 - #define [GAIA_CM](#) 3
 - Length unit conversion: Centimeter.*
 - #define [GAIA_MM](#) 4
 - Length unit conversion: Millimeter.*
 - #define [GAIA_KMI](#) 5
 - Length unit conversion: International Nautical Mile.*
 - #define [GAIA_IN](#) 6
 - Length unit conversion: Inch.*
 - #define [GAIA_FT](#) 7
 - Length unit conversion: Feet.*
 - #define [GAIA_YD](#) 8
 - Length unit conversion: Yard.*
 - #define [GAIA_MI](#) 9
 - Length unit conversion: Mile.*
 - #define [GAIA_FATH](#) 10
 - Length unit conversion: Fathom.*
 - #define [GAIA_CH](#) 11
 - Length unit conversion: Chain.*
 - #define [GAIA_LINK](#) 12
 - Length unit conversion: Link.*
 - #define [GAIA_US_IN](#) 13
 - Length unit conversion: US Inch.*
 - #define [GAIA_US_FT](#) 14
 - Length unit conversion: US Feet.*
 - #define [GAIA_US_YD](#) 15
 - Length unit conversion: US Yard.*
 - #define [GAIA_US_CH](#) 16
 - Length unit conversion: US Chain.*
 - #define [GAIA_US_MI](#) 17
 - Length unit conversion: US Mile.*
 - #define [GAIA_IND_YD](#) 18
 - Length unit conversion: Indian Yard.*
 - #define [GAIA_IND_FT](#) 19
 - Length unit conversion: Indian Feet.*

- #define `GAIA_IND_CH` 20
Length unit conversion: Indian Chain.
- #define `GAIA_MIN_UNIT` `GAIA_KM`
Length unit conversion: MIN.
- #define `GAIA_MAX_UNIT` `GAIA_IND_CH`
Length unit conversion: MAX.
- #define `GAIA_SHP_NULL` 0
SHP shape: unknown.
- #define `GAIA_SHP_POINT` 1
SHP shape: POINT.
- #define `GAIA_SHP_POLYLINE` 3
SHP shape: POLYLINE.
- #define `GAIA_SHP_POLYGON` 5
SHP shape: POLYGON.
- #define `GAIA_SHP_MULTIPPOINT` 8
SHP shape: MULTIPOINT.
- #define `GAIA_SHP_POINTZ` 11
SHP shape: POINT Z.
- #define `GAIA_SHP_POLYLINEZ` 13
SHP shape: POLYLINE Z.
- #define `GAIA_SHP_POLYGONZ` 15
SHP shape: POLYGON Z.
- #define `GAIA_SHP_MULTIPPOINTZ` 18
SHP shape: MULTIPOINT Z.
- #define `GAIA_SHP_POINTM` 21
SHP shape: POINT M.
- #define `GAIA_SHP_POLYLINEM` 23
SHP shape: POLYLINE M.
- #define `GAIA_SHP_POLYGONM` 25
SHP shape: POLYGON M.
- #define `GAIA_SHP_MULTIPPOINTM` 28
SHP shape: MULTIPOINT M.
- #define `GAIA_SAME_ORDER` 0
Clone Special Mode: Same Order as input.
- #define `GAIA_REVERSE_ORDER` -1
Clone Special Mode: Reversed Order.
- #define `GAIA_CW_ORDER` -2
Clone Special Mode: apply Clockwise Rule to Polygon Rings.
- #define `GAIA_CCW_ORDER` -3
Clone Special Mode: apply Counter-Clockwise Rule to Polygon Rings.
- #define `GAIA_DBF_COLNAME_CASE_IGNORE` 0
Leave all DBF column-names as they are.
- #define `GAIA_DBF_COLNAME_LOWERCASE` 1
Convert all DBF column names to LowerCase.
- #define `GAIA_DBF_COLNAME_UPPERCASE` 2
Convert all DBF column names to UpperCase.
- #define `GAIA_PROJ_WKT_ISO_2018` 1
WKT style is ISO-2018.
- #define `GAIA_PROJ_WKT_ISO_2015` 2
WKT style is ISO-2015.
- #define `GAIA_PROJ_WKT_GDAL` 3

- WKT style is GDAL.*

 - #define `GAIA_PROJ_WKT_ESRI` 4

WKT style is ESRI.
- #define `gaiaGetPoint`(xy, v, x, y)

macro extracting XY coordinates
- #define `gaiaSetPoint`(xy, v, x, y)

macro setting XY coordinates
- #define `gaiaGetPointXYZ`(xyz, v, x, y, z)

macro extracting XYZ coordinates
- #define `gaiaSetPointXYZ`(xyz, v, x, y, z)

macro setting XYZ coordinates
- #define `gaiaGetPointXYM`(xym, v, x, y, m)

macro extracting XYM coordinates
- #define `gaiaSetPointXYM`(xym, v, x, y, m)

macro setting XYM coordinates
- #define `gaiaGetPointXYZM`(xyzm, v, x, y, z, m)

macro extracting XYZM coordinates
- #define `gaiaSetPointXYZM`(xyzm, v, x, y, z, m)

macro setting XYZM coordinates

6.6.1 Detailed Description

Geometry constants and macros.

6.6.2 Macro Definition Documentation

6.6.2.1 `gaiaGetPoint`

```
#define gaiaGetPoint(
    xy,
    v,
    x,
    y )
```

Value:

```
{ *x = xy[ (v) * 2]; \
  *y = xy[ (v) * 2 + 1]; }
```

macro extracting XY coordinates

Parameters

<code>xy</code>	pointer [const void *] to COORD mem-array
<code>v</code>	[int] point index [first point has index 0]
<code>x</code>	[double *] X coordinate
<code>y</code>	[double *] Y coordinate

See also

[gaiaLineGetPoint](#), [gaiaRingGetPoint](#)

Note

using this macro on behalf of COORDs not of [XY] dims may cause serious problems

Examples

[demo2.c](#).

6.6.2.2 gaiaGetPointXYM

```
#define gaiaGetPointXYM(  
    xym,  
    v,  
    x,  
    y,  
    m )
```

Value:

```
{ *x = xym[ (v) * 3]; \  
  *y = xym[ (v) * 3 + 1]; \  
  *m = xym[ (v) * 3 + 2]; }
```

macro extracting XYM coordinates

Parameters

<i>xym</i>	pointer [const void *] to COORD mem-array
<i>v</i>	[int] point index [first point has index 0]
<i>x</i>	[double *] X coordinate
<i>y</i>	[double *] Y coordinate
<i>m</i>	[double *] M measure

See also

[gaiaLineGetPoint](#), [gaiaRingGetPoint](#)

Note

using this macro on behalf of COORDs not of [XYM] dims may cause serious problems

6.6.2.3 gaiaGetPointXYZ

```
#define gaiaGetPointXYZ(  
    xyz,  
    v,  
    x,  
    y,  
    z )
```

Value:

```
{ *x = xyz[ (v) * 3]; \  
  *y = xyz[ (v) * 3 + 1]; \  
  *z = xyz[ (v) * 3 + 2]; }
```

macro extracting XYZ coordinates

Parameters

<i>xyz</i>	pointer [const void *] to COORD mem-array
<i>v</i>	[int] point index [first point has index 0]
<i>x</i>	[double *] X coordinate
<i>y</i>	[double *] Y coordinate
<i>z</i>	[double *] Z coordinate

See also

[gaiaLineGetPoint](#), [gaiaRingGetPoint](#)

Note

using this macro on behalf of COORDs not of [XYZ] dims may cause serious problems

6.6.2.4 gaiaGetPointXYZM

```
#define gaiaGetPointXYZM(  
    xyzm,  
    v,  
    x,  
    y,  
    z,  
    m )
```

Value:

```
{ *x = xyzm[ (v) * 4]; \  
  *y = xyzm[ (v) * 4 + 1]; \  
  *z = xyzm[ (v) * 4 + 2]; \  
  *m = xyzm[ (v) * 4 + 3]; }
```

macro extracting XYZM coordinates

Parameters

<i>xyzm</i>	pointer [const void *] to COORD mem-array
<i>v</i>	[int] point index [first point has index 0]
<i>x</i>	[double *] X coordinate
<i>y</i>	[double *] Y coordinate
<i>z</i>	[double *] Z coordinate
<i>m</i>	[double *] M measure

See also

[gaiaLineGetPoint](#), [gaiaRingGetPoint](#)

Note

using this macro on behalf of COORDs not of [XYZM] dims may cause serious problems

6.6.2.5 gaiaSetPoint

```
#define gaiaSetPoint(
    xy,
    v,
    x,
    y )
```

Value:

```
{xy[(v) * 2] = x; \
 xy[(v) * 2 + 1] = y;}
```

macro setting XY coordinates

Parameters

<i>xy</i>	pointer [const void *] to COORD mem-array
<i>v</i>	[int] point index [first point has index 0]
<i>x</i>	[double] X coordinate
<i>y</i>	[double] Y coordinate

See also

[gaiaLineSetPoint](#), [gaiaRingSetPoint](#)

Note

using this macro on behalf on COORDs not of [XY] dims may cause serious problems

Examples

[demo2.c](#).

6.6.2.6 gaiaSetPointXYM

```
#define gaiaSetPointXYM(  
    xym,  
    v,  
    x,  
    y,  
    m )
```

Value:

```
{ xym[ (v) * 3] = x; \  
  xym[ (v) * 3 + 1] = y; \  
  xym[ (v) * 3 + 2] = m; }
```

macro setting XYM coordinates

Parameters

<i>xym</i>	pointer [const void *] to COORD mem-array
<i>v</i>	[int] point index [first point has index 0]
<i>x</i>	[double] X coordinate
<i>y</i>	[double] Y coordinate
<i>m</i>	[double] M measure

See also

[gaiaLineSetPoint](#), [gaiaRingSetPoint](#)

Note

using this macro on behalf on COORDs not of [XYM] dims may cause serious problems

6.6.2.7 gaiaSetPointXYZ

```
#define gaiaSetPointXYZ(  
    xyz,  
    v,  
    x,  
    y,  
    z )
```

Value:

```
{ xyz[ (v) * 3] = x; \  
  xyz[ (v) * 3 + 1] = y; \  
  xyz[ (v) * 3 + 2] = z; }
```

macro setting XYZ coordinates

Parameters

<i>xyz</i>	pointer [const void *] to COORD mem-array
<i>v</i>	[int] point index [first point has index 0]
<i>x</i>	[double] X coordinate
<i>y</i>	[double] Y coordinate
<i>z</i>	[double] Z coordinate

See also

[gaiaLineSetPoint](#), [gaiaRingSetPoint](#)

Note

using this macro on behalf on COORDs not of [XYZ] dims may cause serious problems

6.6.2.8 gaiaSetPointXYZM

```
#define gaiaSetPointXYZM(  
    xyzm,  
    v,  
    x,  
    y,  
    z,  
    m )
```

Value:

```
{xyzm[ (v) * 4] = x; \  
 xyzm[ (v) * 4 + 1] = y; \  
 xyzm[ (v) * 4 + 2] = z; \  
 xyzm[ (v) * 4 + 3] = m; }
```

macro setting XYZM coordinates

Parameters

<i>xyzm</i>	pointer [const void *] to COORD mem-array
<i>v</i>	[int] point index [first point has index 0]
<i>x</i>	[double] X coordinate
<i>y</i>	[double] Y coordinate
<i>z</i>	[double] Z coordinate
<i>m</i>	[double] M measure

See also

[gaiaLineSetPoint](#), [gaiaRingSetPoint](#)

Note

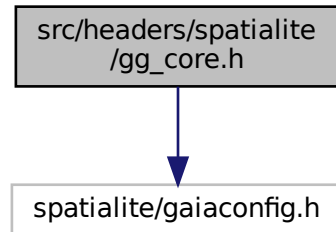
using this macro on behalf on COORDs not of [XYZM] dims may cause serious problems

6.7 src/headers/spatialite/gg_core.h File Reference

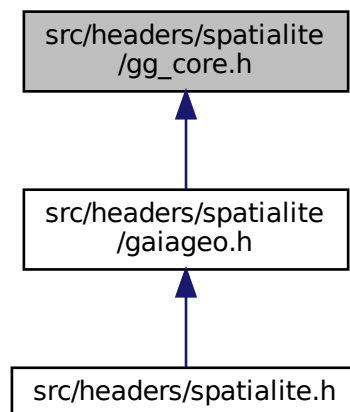
Geometry handling functions: core.

```
#include <spatialite/gaiageo.h>
```

Include dependency graph for gg_core.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define GAIA_GEODESIC_ARC_LENGTH_DEGREES 0`
Arc Length measured in Degrees.
- `#define GAIA_GEODESIC_ARC_LENGTH_METERS 1`
Arc Length measured in Meters.
- `#define GAIA_GEODESIC_CHORD_LENGTH_DEGREES 2`
Chord Length measured in Degrees.
- `#define GAIA_GEODESIC_CHORD_LENGTH_METERS 3`
Chord Length measured in Meters.
- `#define GAIA_GEODESIC_CENTRAL_ANGLE_RADIANS 4`
Central Angle measured in Radians.

- #define [GAIA_GEODESIC_CENTRAL_ANGLE_DEGREES](#) 5
Central Angle measured in Degrees.
- #define [GAIA_GEODESIC_ARC_AREA_METERS](#) 6
Area of segment/arc measured in Square Meters.
- #define [GAIA_GEODESIC_ARC_HEIGHT_METERS](#) 7
Height of segment/arc in Meters.

Functions

- GAIAGEO_DECLARE void [gaiaFree](#) (void *ptr)
Safely frees any dynamic memory block allocated by the library itself.
- GAIAGEO_DECLARE [gaiaPointPtr](#) [gaiaAllocPoint](#) (double x, double y)
Allocates a 2D POINT [XY].
- GAIAGEO_DECLARE [gaiaPointPtr](#) [gaiaAllocPointXYZ](#) (double x, double y, double z)
Allocates a 3D POINT [XYZ].
- GAIAGEO_DECLARE [gaiaPointPtr](#) [gaiaAllocPointXYM](#) (double x, double y, double m)
Allocates a 2D POINT [XYM].
- GAIAGEO_DECLARE [gaiaPointPtr](#) [gaiaAllocPointXYZM](#) (double x, double y, double z, double m)
Allocates a 3D POINT [XYZM].
- GAIAGEO_DECLARE void [gaiaFreePoint](#) ([gaiaPointPtr](#) ptr)
Destroys a POINT object.
- GAIAGEO_DECLARE [gaiaLinestringPtr](#) [gaiaAllocLinestring](#) (int vert)
Allocates a 2D LINESTRING [XY].
- GAIAGEO_DECLARE [gaiaLinestringPtr](#) [gaiaAllocLinestringXYZ](#) (int vert)
Allocates a 3D LINESTRING [XYZ].
- GAIAGEO_DECLARE [gaiaLinestringPtr](#) [gaiaAllocLinestringXYM](#) (int vert)
Allocates a 2D LINESTRING [XYM].
- GAIAGEO_DECLARE [gaiaLinestringPtr](#) [gaiaAllocLinestringXYZM](#) (int vert)
Allocates a 3D LINESTRING [XYZM].
- GAIAGEO_DECLARE void [gaiaFreeLinestring](#) ([gaiaLinestringPtr](#) ptr)
Destroys a LINESTRING object.
- GAIAGEO_DECLARE void [gaiaCopyLinestringCoords](#) ([gaiaLinestringPtr](#) dst, [gaiaLinestringPtr](#) src)
Copies coordinates between two LINESTRING objects.
- GAIAGEO_DECLARE void [gaiaCopyLinestringCoordsEx](#) ([gaiaLinestringPtr](#) dst, [gaiaLinestringPtr](#) src, double z_no_data, double m_no_data)
Copies coordinates between two LINESTRING objects.
- GAIAGEO_DECLARE void [gaiaCopyLinestringCoordsReverse](#) ([gaiaLinestringPtr](#) dst, [gaiaLinestringPtr](#) src)
Copies coordinates between two LINESTRING objects in reverse order.
- GAIAGEO_DECLARE [gaiaRingPtr](#) [gaiaAllocRing](#) (int vert)
Allocates a 2D RING [XY].
- GAIAGEO_DECLARE [gaiaRingPtr](#) [gaiaAllocRingXYZ](#) (int vert)
Allocates a 3D RING [XYZ].
- GAIAGEO_DECLARE [gaiaRingPtr](#) [gaiaAllocRingXYM](#) (int vert)
Allocates 2D RING [XYM].
- GAIAGEO_DECLARE [gaiaRingPtr](#) [gaiaAllocRingXYZM](#) (int vert)
Allocates a 3D RING [XYZM].
- GAIAGEO_DECLARE void [gaiaFreeRing](#) ([gaiaRingPtr](#) ptr)
Destroys a RING object.
- GAIAGEO_DECLARE void [gaiaCopyRingCoords](#) ([gaiaRingPtr](#) dst, [gaiaRingPtr](#) src)
Copies coordinates between two RING objects.

- GAIAGEO_DECLARE void [gaiaCopyRingCoordsEx](#) ([gaiaRingPtr](#) dst, [gaiaRingPtr](#) src, double z_no_data, double m_no_data)
Copies coordinates between two RING objects.
- GAIAGEO_DECLARE void [gaiaCopyRingCoordsReverse](#) ([gaiaRingPtr](#) dst, [gaiaRingPtr](#) src)
Copies coordinates between two RING objects in reverse order.
- GAIAGEO_DECLARE [gaiaPolygonPtr](#) [gaiaAllocPolygon](#) (int vert, int holes)
Allocates a 2D POLYGON [XY].
- GAIAGEO_DECLARE [gaiaPolygonPtr](#) [gaiaAllocPolygonXYZ](#) (int vert, int holes)
Allocates a 3D POLYGON [XYZ].
- GAIAGEO_DECLARE [gaiaPolygonPtr](#) [gaiaAllocPolygonXYM](#) (int vert, int holes)
Allocates a 2D POLYGON [XYM].
- GAIAGEO_DECLARE [gaiaPolygonPtr](#) [gaiaAllocPolygonXYZM](#) (int vert, int holes)
Allocates a 3D POLYGON [XYZM].
- GAIAGEO_DECLARE [gaiaPolygonPtr](#) [gaiaCreatePolygon](#) ([gaiaRingPtr](#) ring)
Allocates a POLYGON.
- GAIAGEO_DECLARE void [gaiaFreePolygon](#) ([gaiaPolygonPtr](#) polyg)
Destroys a POLYGON object.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaAllocGeomColl](#) (void)
Allocates a 2D Geometry [XY].
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaAllocGeomCollXYZ](#) (void)
Allocates a 3D Geometry [XYZ].
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaAllocGeomCollXYM](#) (void)
Allocates a 2D Geometry [XYM].
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaAllocGeomCollXYZM](#) (void)
Allocates a 3D Geometry [XYZM].
- GAIAGEO_DECLARE void [gaiaFreeGeomColl](#) ([gaiaGeomCollPtr](#) geom)
Destroys a Geometry object.
- GAIAGEO_DECLARE void [gaiaAddPointToGeomColl](#) ([gaiaGeomCollPtr](#) p, double x, double y)
Creates a new 2D Point [XY] object into a Geometry object.
- GAIAGEO_DECLARE void [gaiaAddPointToGeomCollXYZ](#) ([gaiaGeomCollPtr](#) p, double x, double y, double z)
Creates a new 3D Point [XYZ] object into a Geometry object.
- GAIAGEO_DECLARE void [gaiaAddPointToGeomCollXYM](#) ([gaiaGeomCollPtr](#) p, double x, double y, double m)
Creates a new 2D Point [XYM] object into a Geometry object.
- GAIAGEO_DECLARE void [gaiaAddPointToGeomCollXYZM](#) ([gaiaGeomCollPtr](#) p, double x, double y, double z, double m)
Creates a new 3D Point [XYZM] object into a Geometry object.
- GAIAGEO_DECLARE [gaiaLinestringPtr](#) [gaiaAddLinestringToGeomColl](#) ([gaiaGeomCollPtr](#) p, int vert)
Creates a new Linestring object into a Geometry object.
- GAIAGEO_DECLARE void [gaiaInsertLinestringInGeomColl](#) ([gaiaGeomCollPtr](#) p, [gaiaLinestringPtr](#) line)
Inserts an already existing Linestring object into a Geometry object.
- GAIAGEO_DECLARE [gaiaPolygonPtr](#) [gaiaAddPolygonToGeomColl](#) ([gaiaGeomCollPtr](#) p, int vert, int interiors)
Creates a new Polygon object into a Geometry object.
- GAIAGEO_DECLARE [gaiaPolygonPtr](#) [gaiaInsertPolygonInGeomColl](#) ([gaiaGeomCollPtr](#) p, [gaiaRingPtr](#) ring)
Creates a new Polygon object into a Geometry object starting from an already existing Ring object.
- GAIAGEO_DECLARE [gaiaRingPtr](#) [gaiaAddInteriorRing](#) ([gaiaPolygonPtr](#) p, int pos, int vert)
Creates a new Interior Ring object into a Polygon object.
- GAIAGEO_DECLARE void [gaiaInsertInteriorRing](#) ([gaiaPolygonPtr](#) p, [gaiaRingPtr](#) ring)
Inserts an already existing Ring object into a Polygon object.
- GAIAGEO_DECLARE void [gaiaAddRingToPolyg](#) ([gaiaPolygonPtr](#) polyg, [gaiaRingPtr](#) ring)
Inserts an already existing Ring object into a Polygon object.

- GAIAGEO_DECLARE [gaiaLinestringPtr](#) [gaiaCloneLinestring](#) ([gaiaLinestringPtr](#) line)
Duplicates a Linestring object.
- GAIAGEO_DECLARE [gaiaLinestringPtr](#) [gaiaCloneLinestringSpecial](#) ([gaiaLinestringPtr](#) line, int mode)
Duplicates a Linestring object (special)
- GAIAGEO_DECLARE [gaiaRingPtr](#) [gaiaCloneRing](#) ([gaiaRingPtr](#) ring)
Duplicates a Ring object.
- GAIAGEO_DECLARE [gaiaRingPtr](#) [gaiaCloneRingSpecial](#) ([gaiaRingPtr](#) ring, int mode)
Duplicates a Ring object (special)
- GAIAGEO_DECLARE [gaiaPolygonPtr](#) [gaiaClonePolygon](#) ([gaiaPolygonPtr](#) polyg)
Duplicates a Polygon object.
- GAIAGEO_DECLARE [gaiaPolygonPtr](#) [gaiaClonePolygonSpecial](#) ([gaiaPolygonPtr](#) polyg, int mode)
Duplicates a Polygon object (special)
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaCloneGeomColl](#) ([gaiaGeomCollPtr](#) geom)
Duplicates a Geometry object.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaCloneGeomCollSpecial](#) ([gaiaGeomCollPtr](#) geom, int mode)
Duplicates a Geometry object (special)
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaCloneGeomCollPoints](#) ([gaiaGeomCollPtr](#) geom)
Duplicates a Geometry object [Points only].
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaCloneGeomCollLinestrings](#) ([gaiaGeomCollPtr](#) geom)
Duplicates a Geometry object [Linestrings only].
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaCloneGeomCollPolygons](#) ([gaiaGeomCollPtr](#) geom)
Duplicates a Geometry object [Polygons only].
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaCastGeomCollToXY](#) ([gaiaGeomCollPtr](#) geom)
Duplicates a Geometry object [casting dimensions to 2D XY].
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaCastGeomCollToXYZ](#) ([gaiaGeomCollPtr](#) geom)
Duplicates a Geometry object [casting dimensions to 3D XYZ].
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaCastGeomCollToXYM](#) ([gaiaGeomCollPtr](#) geom)
Duplicates a Geometry object [casting dimensions to 2D XYM].
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaCastGeomCollToXYZM](#) ([gaiaGeomCollPtr](#) geom)
Duplicates a Geometry object [casting dimensions to 3D XYZM].
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaCastGeomCollToXYZnoData](#) ([gaiaGeomCollPtr](#) geom, double no_data)
Duplicates a Geometry object [casting dimensions to 3D XYZ - noData].
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaCastGeomCollToXYMnoData](#) ([gaiaGeomCollPtr](#) geom, double no_data)
Duplicates a Geometry object [casting dimensions to 2D XYM - noData].
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaCastGeomCollToXYZMnoData](#) ([gaiaGeomCollPtr](#) geom, double z_no_data, double m_no_data)
Duplicates a Geometry object [casting dimensions to 3D XYZM - noData].
- GAIAGEO_DECLARE int [gaiaLineGetPoint](#) ([gaiaLinestringPtr](#) ln, int v, double *x, double *y, double *z, double *m)
Gets coordinates from a Linestring's Point.
- GAIAGEO_DECLARE int [gaiaLineSetPoint](#) ([gaiaLinestringPtr](#) ln, int v, double x, double y, double z, double m)
Sets coordinates for a Linestring's Point.
- GAIAGEO_DECLARE int [gaiaRingGetPoint](#) ([gaiaRingPtr](#) rng, int v, double *x, double *y, double *z, double *m)
Gets coordinates from a Ring's Point.
- GAIAGEO_DECLARE int [gaiaRingSetPoint](#) ([gaiaRingPtr](#) rng, int v, double x, double y, double z, double m)
Sets coordinates for a Ring's Point.
- GAIAGEO_DECLARE int [gaiaDimension](#) ([gaiaGeomCollPtr](#) geom)

- Determines OGC dimensions for a Geometry object.*
- GAIA GEO_DECLARE int [gaiaGeometryType](#) ([gaiaGeomCollPtr](#) geom)
- Determines the corresponding Type for a Geometry object.*
- GAIA GEO_DECLARE int [gaiaGeometryAliasType](#) ([gaiaGeomCollPtr](#) geom)
- Determines the corresponding Type for a Geometry object.*
- GAIA GEO_DECLARE int [gaiaIsEmpty](#) ([gaiaGeomCollPtr](#) geom)
- Checks for empty Geometry object.*
- GAIA GEO_DECLARE int [gaiaCheckClockwise](#) ([gaiaGeomCollPtr](#) geom)
- Checks for Clockwise Geometry object.*
- GAIA GEO_DECLARE int [gaiaCheckCounterClockwise](#) ([gaiaGeomCollPtr](#) geom)
- Checks for CounterClockwise Geometry object.*
- GAIA GEO_DECLARE int [gaiaIsToxic](#) ([gaiaGeomCollPtr](#) geom)
- Checks for toxic Geometry object.*
- GAIA GEO_DECLARE int [gaiaIsToxic_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom)
- Checks for toxic Geometry object.*
- GAIA GEO_DECLARE int [gaiaIsNotClosedRing](#) ([gaiaRingPtr](#) ring)
- Checks for not-closed Rings.*
- GAIA GEO_DECLARE int [gaiaIsNotClosedRing_r](#) (const void *p_data, [gaiaRingPtr](#) ring)
- Checks for not-closed Rings.*
- GAIA GEO_DECLARE int [gaiaIsNotClosedGeomColl](#) ([gaiaGeomCollPtr](#) geom)
- Checks for not-closed Rings in a Geometry object.*
- GAIA GEO_DECLARE int [gaiaIsNotClosedGeomColl_r](#) (const void *p_data, [gaiaGeomCollPtr](#) geom)
- Checks for not-closed Rings in a Geometry object.*
- GAIA GEO_DECLARE [gaiaGeomCollPtr](#) [gaiaSanitize](#) ([gaiaGeomCollPtr](#) org)
- Attempts to sanitize a possibly malformed Geometry object.*
- GAIA GEO_DECLARE [gaiaGeomCollPtr](#) [gaiaEnsureClosedRings](#) ([gaiaGeomCollPtr](#) org)
- Attempts to sanitize a possibly malformed Geometry object.*
- GAIA GEO_DECLARE [gaiaGeomCollPtr](#) [gaiaRemoveRepeatedPoints](#) ([gaiaGeomCollPtr](#) org, double tolerance)
- Attempts to sanitize a possibly malformed Geometry object.*
- GAIA GEO_DECLARE [gaiaGeomCollPtr](#) [gaiaLinearize](#) ([gaiaGeomCollPtr](#) geom, int force_multi)
- Attempts to resolve a (Multi)Linestring from a Geometry object.*
- GAIA GEO_DECLARE [gaiaGeomCollPtr](#) [gaiaDissolveSegments](#) ([gaiaGeomCollPtr](#) geom)
- Attempts to resolve a collection of Segments from a Geometry object.*
- GAIA GEO_DECLARE [gaiaGeomCollPtr](#) [gaiaDissolvePoints](#) ([gaiaGeomCollPtr](#) geom)
- Attempts to resolve a collection of Points from a Geometry object.*
- GAIA GEO_DECLARE [gaiaGeomCollPtr](#) [gaiaExtractPointsFromGeomColl](#) ([gaiaGeomCollPtr](#) geom)
- Extracts any Point from a Geometry object.*
- GAIA GEO_DECLARE [gaiaGeomCollPtr](#) [gaiaExtractLinestringsFromGeomColl](#) ([gaiaGeomCollPtr](#) geom)
- Extracts any Linestring from a Geometry object.*
- GAIA GEO_DECLARE [gaiaGeomCollPtr](#) [gaiaExtractPolygonsFromGeomColl](#) ([gaiaGeomCollPtr](#) geom)
- Extracts any Polygon from a Geometry object.*
- GAIA GEO_DECLARE [gaiaGeomCollPtr](#) [gaiaMergeGeometries](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
- Merges two Geometry objects into a single one.*
- GAIA GEO_DECLARE [gaiaGeomCollPtr](#) [gaiaMergeGeometries_r](#) (const void *p_cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2)
- Merges two Geometry objects into a single one.*
- GAIA GEO_DECLARE [gaiaGeomCollPtr](#) [gaiaAddMeasure](#) ([gaiaGeomCollPtr](#) geom, double m_start, double m_end)
- Will return a new GEOMETRY (supporting M) with measures linearly interpolated between the start and end points.*

- GAIAGEO_DECLARE int [gaiaInterpolatePoint](#) (const void *p_cache, [gaiaGeomCollPtr](#) line, [gaiaGeomCollPtr](#) point, double *m_value)
Will interpolate the M-value for a LinestringM at the point closest to the given Point.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaLocateBetweenMeasures](#) ([gaiaGeomCollPtr](#) geom, double m_start, double m_end)
Return a GeometryCollection containing elements matching the specified range of measures.
- GAIAGEO_DECLARE int [gaiaIsValidTrajectory](#) ([gaiaGeomCollPtr](#) geom)
Checks if a Geometry object is valid Trajectory.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaTrajectoryInterpolatePoint](#) ([gaiaGeomCollPtr](#) geom, double m)
Attempts to interpolate a Point along a Trajectory accordingly to given M-Value.
- GAIAGEO_DECLARE double [gaiaMeasureLength](#) (int dims, double *coords, int vert)
Measures the geometric length for a Linestring or Ring.
- GAIAGEO_DECLARE double [gaiaMeasureArea](#) ([gaiaRingPtr](#) ring)
Measures the geometric area for a Ring object.
- GAIAGEO_DECLARE void [gaiaRingCentroid](#) ([gaiaRingPtr](#) ring, double *rx, double *ry)
Determines the Centroid for a Ring object.
- GAIAGEO_DECLARE void [gaiaClockwise](#) ([gaiaRingPtr](#) p)
Determines the direction for a Ring object.
- GAIAGEO_DECLARE int [gaiaIsPointOnRingSurface](#) ([gaiaRingPtr](#) ring, double pt_x, double pt_y)
Check if a Point lays on a Ring surface.
- GAIAGEO_DECLARE int [gaiaIsPointOnPolygonSurface](#) ([gaiaPolygonPtr](#) polyg, double x, double y)
Checks if a Point lays on a Polygon surface.
- GAIAGEO_DECLARE double [gaiaMinDistance](#) (double x0, double y0, int dims, double *coords, int vert)
Computes the minimum distance between a Point and a Linestring or Ring.
- GAIAGEO_DECLARE int [gaiaIntersect](#) (double *x0, double *y0, double x1, double y1, double x2, double y2, double x3, double y3, double x4, double y4)
Determines the intersection Point between two Segments.
- GAIAGEO_DECLARE void [gaiaShiftCoords](#) ([gaiaGeomCollPtr](#) geom, double shift_x, double shift_y)
Shifts any coordinate within a Geometry object.
- GAIAGEO_DECLARE void [gaiaShiftCoords3D](#) ([gaiaGeomCollPtr](#) geom, double shift_x, double shift_y, double shift_z)
Shifts any coordinate within a 3D Geometry object.
- GAIAGEO_DECLARE void [gaiaShiftLongitude](#) ([gaiaGeomCollPtr](#) geom)
Shifts negative longitudes.
- GAIAGEO_DECLARE void [gaiaNormalizeLonLat](#) ([gaiaGeomCollPtr](#) geom)
Shifts any coordinate to within the "normal range" of longitude and latitude values (-180.0 to 180.0 longitude and -90.0 to 90.0 latitude).
- GAIAGEO_DECLARE void [gaiaScaleCoords](#) ([gaiaGeomCollPtr](#) geom, double scale_x, double scale_y)
Scales any coordinate within a Geometry object.
- GAIAGEO_DECLARE void [gaiaRotateCoords](#) ([gaiaGeomCollPtr](#) geom, double angle)
Rotates any coordinate within a Geometry object.
- GAIAGEO_DECLARE void [gaiaReflectCoords](#) ([gaiaGeomCollPtr](#) geom, int x_axis, int y_axis)
Reflects any coordinate within a Geometry object.
- GAIAGEO_DECLARE void [gaiaSwapCoords](#) ([gaiaGeomCollPtr](#) geom)
Swaps any coordinate within a Geometry object.
- GAIAGEO_DECLARE int [gaiaLinestringEquals](#) ([gaiaLinestringPtr](#) line1, [gaiaLinestringPtr](#) line2)
Checks if two Linestring objects are equivalent.
- GAIAGEO_DECLARE int [gaiaPolygonEquals](#) ([gaiaPolygonPtr](#) polyg1, [gaiaPolygonPtr](#) polyg2)
Checks if two Polygons objects are equivalent.
- GAIAGEO_DECLARE int [gaiaEllipseParams](#) (const char *name, double *a, double *b, double *rf)
Retrieves Geodesic params for an Ellipsoid definition.

- GAIAGEO_DECLARE double [gaiaGreatCircleDistance](#) (double a, double b, double lat1, double lon1, double lat2, double lon2)
Calculates the Great Circle Distance between two Points.
- GAIAGEO_DECLARE double [gaiaGeodesicDistance](#) (double a, double b, double rf, double lat1, double lon1, double lat2, double lon2)
Calculates the Geodesic Distance between two Points.
- GAIAGEO_DECLARE double [gaiaGreatCircleTotalLength](#) (double a, double b, int dims, double *coords, int vert)
Calculates the Great Circle Total Length for a Linestring / Ring.
- GAIAGEO_DECLARE double [gaiaGeodesicTotalLength](#) (double a, double b, double rf, int dims, double *coords, int vert)
Calculates the Geodesic Total Length for a Linestring / Ring.
- GAIAGEO_DECLARE int [gaiaConvertLength](#) (double value, int unit_from, int unit_to, double *cvt)
Convert a Length from a Measure Unit to another.
- GAIAGEO_DECLARE int [gaiaGeodesicArcLength](#) (sqlite3 *sqlite, const void *cache, [gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, int return_type, double *retval)
Computes several Geodesic values based on the Distance between two Geometries.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaMakeCircle](#) (double center_x, double center_y, double radius, double step)
Creates a Circle (Linestring) Geometry.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaMakeEllipse](#) (double center_x, double center_y, double x_axis, double y_axis, double step)
Creates an Ellipse (Linestring) Geometry.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaMakeArc](#) (double center_x, double center_y, double radius, double start, double stop, double step)
Creates a Circular Arc (Linestring) Geometry.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaMakeEllipticArc](#) (double center_x, double center_y, double x_axis, double y_axis, double start, double stop, double step)
Creates an Elliptic Arc (Linestring) Geometry.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaMakePolygon](#) ([gaiaGeomCollPtr](#) exterior, [gaiaGeomCollPtr](#) interiors)
Creates a Polygon from closed Linestrings.
- GAIAGEO_DECLARE double [gaiaCurvosityIndex](#) (const void *p_cache, [gaiaLinestringPtr](#) line, int extra_points)
Computes the Curvosity Index for some Linestrings.
- GAIAGEO_DECLARE void [gaiaUpDownHeight](#) ([gaiaLinestringPtr](#) line, double *up, double *down)
Computes the Uphill and Downhill total Height for some 3D Linestrings.

6.7.1 Detailed Description

Geometry handling functions: core.

6.7.2 Function Documentation

6.7.2.1 [gaiaAddInteriorRing\(\)](#)

```
GAIAGEO_DECLARE gaiaRingPtr gaiaAddInteriorRing (
    gaiaPolygonPtr p,
    int pos,
    int vert )
```

Creates a new Interior Ring object into a Polygon object.

Parameters

<i>p</i>	pointer to the Polygon object.
<i>pos</i>	relative position index [first Interior Ring has index 0].
<i>vert</i>	number of points (aka vertices) into the Ring.

Returns

the pointer to the newly created Ring object: NULL on failure.

See also

[gaiaAllocPolygon](#), [gaiaAllocPolygonXYZ](#), [gaiaAllocPolygonXYM](#), [gaiaAllocPolygonXYZM](#)

Note

ownership of the Ring object belongs to the Polygon object.
the newly created Ring will have the same dimensions the Polygon has.

Examples

[demo2.c](#).

6.7.2.2 gaiaAddLinestringToGeomColl()

```
GAIA GEO_DECLARE gaiaLinestringPtr gaiaAddLinestringToGeomColl (
    gaiaGeomCollPtr p,
    int vert )
```

Creates a new Linestring object into a Geometry object.

Parameters

<i>p</i>	pointer to the Geometry object.
<i>vert</i>	number of points [aka vertices] into the Linestring.

Returns

the pointer to newly created Linestring: NULL on failure.

Note

ownership of the newly created Linestring object belongs to the Geometry object.
the newly created Linestring will have the same dimensions as the Geometry has.

Examples

[demo2.c](#).

6.7.2.3 gaiaAddMeasure()

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaAddMeasure (
    gaiaGeomCollPtr geom,
    double m_start,
    double m_end )
```

Will return a new GEOMETRY (supporting M) with measures linearly interpolated between the start and end points.

Parameters

<i>geom</i>	pointer to Geometry object of the Linestring or MultiLinestring type.
<i>m_start</i>	M start value
<i>m_end</i>	M end value

Returns

the pointer to newly created Geometry: NULL on failure.

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry created by [gaiaAddMeasure\(\)](#)
the newly created Geometry will contain Linestrings.
if the input Geometry has no M dimension it will be added, otherwise it will be overwritten.
an eventual Z will be preserved unaffected.

6.7.2.4 gaiaAddPointToGeomColl()

```
GAIA GEO_DECLARE void gaiaAddPointToGeomColl (
    gaiaGeomCollPtr p,
    double x,
    double y )
```

Creates a new 2D Point [XY] object into a Geometry object.

Parameters

<i>p</i>	pointer to the Geometry object
<i>x</i>	X coordinate of the Point to be created
<i>y</i>	Y coordinate of the Point to be created

Note

ownership of the newly created POINT object belongs to the Geometry object.

Examples

[demo2.c](#), [demo3.c](#), and [demo4.c](#).

6.7.2.5 gaiaAddPointToGeomCollXYM()

```
GAIA GEO_DECLARE void gaiaAddPointToGeomCollXYM (
    gaiaGeomCollPtr p,
    double x,
    double y,
    double m )
```

Creates a new 2D Point [XYM] object into a Geometry object.

Parameters

<i>p</i>	pointer to the Geometry object
<i>x</i>	X coordinate of the Point to be created
<i>y</i>	X coordinate of the Point to be created
<i>m</i>	M measure of the Point to be created

Note

ownership of the newly created POINT object belongs to the Geometry object.

6.7.2.6 gaiaAddPointToGeomCollXYZ()

```
GAIA GEO_DECLARE void gaiaAddPointToGeomCollXYZ (
    gaiaGeomCollPtr p,
    double x,
    double y,
    double z )
```

Creates a new 3D Point [XYZ] object into a Geometry object.

Parameters

<i>p</i>	pointer to the Geometry object
<i>x</i>	X coordinate of the Point to be created
<i>y</i>	X coordinate of the Point to be created
<i>z</i>	Z coordinate of the Point to be created

Note

ownership of the newly created POINT object belongs to the Geometry object.

6.7.2.7 gaiaAddPointToGeomCollXYZM()

```
GAIA GEO_DECLARE void gaiaAddPointToGeomCollXYZM (
    gaiaGeomCollPtr p,
```

```
double x,  
double y,  
double z,  
double m )
```

Creates a new 3D Point [XYZM] object into a Geometry object.

Parameters

<i>p</i>	pointer to the Geometry object
<i>x</i>	X coordinate of the Point to be created
<i>y</i>	X coordinate of the Point to be created
<i>z</i>	Z coordinate of the Point to be created
<i>m</i>	M measure of the Point to be created

Note

ownership of the newly created POINT object belongs to the Geometry object.

6.7.2.8 gaiaAddPolygonToGeomColl()

```
GAIA GEO_DECLARE gaiaPolygonPtr gaiaAddPolygonToGeomColl (  
    gaiaGeomCollPtr p,  
    int vert,  
    int interiors )
```

Creates a new Polygon object into a Geometry object.

Parameters

<i>p</i>	pointer to the Geometry object.
<i>vert</i>	number of points [aka vertices] into the Polygon's Exterior Ring.
<i>interiors</i>	number of Interiors Rings [0, if no Interior Ring is required]

Returns

the pointer to newly created Polygon: NULL on failure.

Note

ownership of the newly created Polygon object belongs to the Geometry object.
the newly created Polygon will have the same dimensions as the Geometry has.

Examples

[demo2.c](#).

6.7.2.9 `gaiaAddRingToPolyg()`

```
GAIAGEO_DECLARE void gaiaAddRingToPolyg (
    gaiaPolygonPtr polyg,
    gaiaRingPtr ring )
```

Inserts an already existing Ring object into a Polygon object.

Parameters

<i>polyg</i>	pointer to the Polygon object
<i>ring</i>	pointer to the Ring object

See also

[gaiaInsertInteriorRing](#)

Note

ownership of the Ring object will be transferred to the Polygon object.
the newly created Polygon will have the same dimensions as the Ring has.
if required the Polygon's Interior Rings count could be increased.

6.7.2.10 `gaiaAllocGeomColl()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaAllocGeomColl (
    void )
```

Allocates a 2D Geometry [XY].

Returns

the pointer to newly created Geometry object: NULL on failure

See also

[gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, unless you've passed ownership of the Geometry object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

Examples

[demo2.c](#), [demo3.c](#), and [demo4.c](#).

6.7.2.11 `gaiaAllocGeomCollXYM()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaAllocGeomCollXYM (  
    void )
```

Allocates a 2D Geometry [XYM].

Returns

the pointer to newly created Geometry object: NULL on failure

See also

[gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, unless you've passed ownership of the Geometry object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.7.2.12 `gaiaAllocGeomCollXYZ()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaAllocGeomCollXYZ (  
    void )
```

Allocates a 3D Geometry [XYZ].

Returns

the pointer to newly created Geometry object: NULL on failure

See also

[gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, unless you've passed ownership of the Geometry object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.7.2.13 `gaiaAllocGeomCollXYZM()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaAllocGeomCollXYZM (  
    void )
```

Allocates a 3D Geometry [XYZM].

Returns

the pointer to newly created Geometry object: NULL on failure

See also

[gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, unless you've passed ownership of the Geometry object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.7.2.14 `gaiaAllocLinestring()`

```
GAIAGEO_DECLARE gaiaLinestringPtr gaiaAllocLinestring (  
    int vert )
```

Allocates a 2D LINESTRING [XY].

Parameters

<i>vert</i>	number of points [aka vertices] into the Linestring
-------------	---

Returns

the pointer to newly created LINESTRING object: NULL on failure

See also

[gaiaFreeLinestring](#), [gaiaLineSetPoint](#), [gaiaLineGetPoint](#), [gaiaSetPoint](#), [gaiaGetPoint](#)

Note

you are responsible to destroy (before or after) any allocated LINESTRING, unless you've passed ownership of the LINESTRING object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.7.2.15 `gaiaAllocLinestringXYM()`

```
GAIAGEO_DECLARE gaiaLinestringPtr gaiaAllocLinestringXYM (  
    int vert )
```

Allocates a 2D LINESTRING [XYM].

Parameters

<i>vert</i>	number of points [aka vertices] into the Linestring
-------------	---

Returns

the pointer to newly created LINESTRING object: NULL on failure

See also

[gaiaFreeLinestring](#), [gaiaLineSetPoint](#), [gaiaLineGetPoint](#), [gaiaSetPointXYM](#), [gaiaGetPointXYM](#)

Note

you are responsible to destroy (before or after) any allocated LINESTRING, unless you've passed ownership of the LINESTRING object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.7.2.16 `gaiaAllocLinestringXYZ()`

```
GAIAGEO_DECLARE gaiaLinestringPtr gaiaAllocLinestringXYZ (  
    int vert )
```

Allocates a 3D LINESTRING [XYZ].

Parameters

<i>vert</i>	number of points [aka vertices] into the Linestring
-------------	---

Returns

the pointer to newly created LINESTRING object: NULL on failure

See also

[gaiaFreeLinestring](#), [gaiaLineSetPoint](#), [gaiaLineGetPoint](#), [gaiaSetPointXYZ](#), [gaiaGetPointXYZ](#)

Note

you are responsible to destroy (before or after) any allocated LINESTRING, unless you've passed ownership of the LINESTRING object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.7.2.17 gaiaAllocLinestringXYZM()

```
GAIAGEO_DECLARE gaiaLinestringPtr gaiaAllocLinestringXYZM (  
    int vert )
```

Allocates a 3D LINESTRING [XYZM].

Parameters

<i>vert</i>	number of points [aka vertices] into the Linestring
-------------	---

Returns

the pointer to newly created LINESTRING object: NULL on failure

See also

[gaiaFreeLinestring](#), [gaiaLineSetPoint](#), [gaiaLineGetPoint](#), [gaiaSetPointXYZM](#), [gaiaGetPointXYZM](#)

Note

you are responsible to destroy (before or after) any allocated LINESTRING, unless you've passed ownership of the LINESTRING object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.7.2.18 gaiaAllocPoint()

```
GAIAGEO_DECLARE gaiaPointPtr gaiaAllocPoint (  
    double x,  
    double y )
```

Allocates a 2D POINT [XY].

Parameters

<i>x</i>	the X coordinate.
<i>y</i>	the Y coordinate.

Returns

the pointer to the newly created POINT object: NULL on failure

See also

[gaiaFreePoint](#)

Note

you are responsible to destroy (before or after) any allocated POINT, unless you've passed ownership of the POINT object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.7.2.19 gaiaAllocPointXYM()

```
GAIA GEO_DECLARE gaiaPointPtr gaiaAllocPointXYM (
    double x,
    double y,
    double m )
```

Allocates a 2D POINT [XYM].

Parameters

<i>x</i>	the X coordinate.
<i>y</i>	the Y coordinate.
<i>m</i>	the M measure.

Returns

the pointer to the newly created POINT object: NULL on failure

See also

[gaiaFreePoint](#)

Note

you are responsible to destroy (before or after) any allocated POINT, unless you've passed ownership of the POINT object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.7.2.20 gaiaAllocPointXYZ()

```
GAIA GEO_DECLARE gaiaPointPtr gaiaAllocPointXYZ (
    double x,
    double y,
    double z )
```

Allocates a 3D POINT [XYZ].

Parameters

<i>x</i>	the X coordinate.
<i>y</i>	the Y coordinate.
<i>z</i>	the Z coordinate.

Returns

the pointer to the newly created POINT object: NULL on failure

See also

[gaiaFreePoint](#)

Note

you are responsible to destroy (before or after) any allocated POINT, unless you've passed ownership of the POINT object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.7.2.21 gaiaAllocPointXYZM()

```
GAIA GEO_DECLARE gaiaPointPtr gaiaAllocPointXYZM (
    double x,
    double y,
    double z,
    double m )
```

Allocates a 3D POINT [XYZM].

Parameters

<i>x</i>	the X coordinate.
<i>y</i>	the Y coordinate.
<i>z</i>	the Z coordinate.
<i>m</i>	the M measure.

Returns

the pointer to the newly created POINT object: NULL on failure

See also

[gaiaFreePoint](#)

Note

you are responsible to destroy (before or after) any allocated POINT, unless you've passed ownership of the POINT object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.7.2.22 gaiaAllocPolygon()

```
GAIA GEO_DECLARE gaiaPolygonPtr gaiaAllocPolygon (
    int vert,
    int holes )
```

Allocates a 2D POLYGON [XY].

Parameters

<i>vert</i>	number of points [aka vertices] into the Exterior Ring.
<i>holes</i>	number of Interior Rings [0, if no Interior Ring is required].

Returns

the pointer to newly created POLYGON object: NULL on failure

See also

[gaiaFreePolygon](#)

Note

you are responsible to destroy (before or after) any allocated POLYGON, unless you've passed ownership of the POLYGON object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.7.2.23 gaiaAllocPolygonXYM()

```
GAIA GEO_DECLARE gaiaPolygonPtr gaiaAllocPolygonXYM (
    int vert,
    int holes )
```

Allocates a 2D POLYGON [XYM].

Parameters

<i>vert</i>	number of points [aka vertices] into the Exterior Ring.
<i>holes</i>	number of Interior Rings [0, if no Interior Ring is required].

Returns

the pointer to newly created POLYGON object: NULL on failure

See also

[gaiaFreePolygon](#)

Note

you are responsible to destroy (before or after) any allocated POLYGON, unless you've passed ownership of the POLYGON object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.7.2.24 gaiaAllocPolygonXYZ()

```
GAIA GEO_DECLARE gaiaPolygonPtr gaiaAllocPolygonXYZ (
    int vert,
    int holes )
```

Allocates a 3D POLYGON [XYZ].

Parameters

<i>vert</i>	number of points [aka vertices] into the Exterior Ring.
<i>holes</i>	number of Interior Rings [0, if no Interior Ring is required].

Returns

the pointer to newly created POLYGON object: NULL on failure

See also

[gaiaFreePolygon](#)

Note

you are responsible to destroy (before or after) any allocated POLYGON, unless you've passed ownership of the POLYGON object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.7.2.25 gaiaAllocPolygonXYZM()

```
GAIA GEO_DECLARE gaiaPolygonPtr gaiaAllocPolygonXYZM (
    int vert,
    int holes )
```

Allocates a 3D POLYGON [XYZM].

Parameters

<i>vert</i>	number of points [aka vertices] into the Exterior Ring.
<i>holes</i>	number of Interior Rings [may by 0, if no Interior Ring is required].

Returns

the pointer to newly created POLYGON object: NULL on failure

See also

[gaiaFreePolygon](#)

Note

you are responsible to destroy (before or after) any allocated POLYGON, unless you've passed ownership of the POLYGON object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.7.2.26 gaiaAllocRing()

```
GAIA GEO_DECLARE gaiaRingPtr gaiaAllocRing (
    int vert )
```

Allocates a 2D RING [XY].

Parameters

<i>vert</i>	number of points [aka vertices] into the Ring
-------------	---

Returns

the pointer to newly created RING object: NULL on failure

See also

[gaiaFreeRing](#), [gaiaRingSetPoint](#), [gaiaRingGetPoint](#), [gaiaSetPoint](#), [gaiaGetPoint](#)

Note

you are responsible to destroy (before or after) any allocated RING, unless you've passed ownership of the RING object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.7.2.27 `gaiaAllocRingXYM()`

```
GAIA GEO_DECLARE gaiaRingPtr gaiaAllocRingXYM (
    int vert )
```

Allocates 2D RING [XYM].

Parameters

<code>vert</code>	number of points [aka vertices] into the Ring
-------------------	---

Returns

the pointer to newly created RING object: NULL on failure

See also

[gaiaFreeRing](#), [gaiaRingSetPoint](#), [gaiaRingGetPoint](#), [gaiaSetPointXYM](#), [gaiaGetPointXYM](#)

Note

you are responsible to destroy (before or after) any allocated RING, unless you've passed ownership of the RING object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.7.2.28 `gaiaAllocRingXYZ()`

```
GAIA GEO_DECLARE gaiaRingPtr gaiaAllocRingXYZ (
    int vert )
```

Allocates a 3D RING [XYZ].

Parameters

<code>vert</code>	number of points [aka vertices] into the Ring
-------------------	---

Returns

the pointer to newly created RING object: NULL on failure

See also

[gaiaFreeRing](#), [gaiaRingSetPoint](#), [gaiaRingGetPoint](#), [gaiaSetPointXYZ](#), [gaiaGetPointXYZ](#)

Note

you are responsible to destroy (before or after) any allocated RING, unless you've passed ownership of the RING object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.7.2.29 gaiaAllocRingXYZM()

```
GAIA GEO_DECLARE gaiaRingPtr gaiaAllocRingXYZM (
    int vert )
```

Allocates a 3D RING [XYZM].

Parameters

<i>vert</i>	number of points [aka vertices] into the Ring
-------------	---

Returns

the pointer to newly created RING object: NULL on failure

See also

[gaiaFreeRing](#), [gaiaRingSetPoint](#), [gaiaRingGetPoint](#), [gaiaSetPointXYZM](#), [gaiaSetPointXYZM](#)

Note

you are responsible to destroy (before or after) any allocated RING, unless you've passed ownership of the RING object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.7.2.30 gaiaCastGeomCollToXY()

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaCastGeomCollToXY (
    gaiaGeomCollPtr geom )
```

Duplicates a Geometry object [casting dimensions to 2D XY].

Parameters

<i>geom</i>	pointer to Geometry object [origin].
-------------	--------------------------------------

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaCloneGeomColl](#), [gaiaCastGeomCollToXYZ](#), [gaiaCastGeomCollToXYM](#), [gaiaCastGeomCollToXYZM](#)

Note

the newly created object is an exact copy of the original one; except in that any elementary item will be casted to 2D [XY] dimensions.

6.7.2.31 [gaiaCastGeomCollToXYM\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaCastGeomCollToXYM (  
    gaiaGeomCollPtr geom )
```

Duplicates a Geometry object [casting dimensions to 2D XYM].

Parameters

<i>geom</i>	pointer to Geometry object [origin].
-------------	--------------------------------------

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaCloneGeomColl](#), [gaiaCastGeomCollToXY](#), [gaiaCastGeomCollToXYZ](#), [gaiaCastGeomCollToXYZM](#), [gaiaCastGeomCollToXYMnoData](#)

Note

the newly created object is an exact copy of the original one; except in that any elementary item will be cast to 2D [XYM] dimensions.

6.7.2.32 [gaiaCastGeomCollToXYMnoData\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaCastGeomCollToXYMnoData (  
    gaiaGeomCollPtr geom,  
    double no_data )
```

Duplicates a Geometry object [casting dimensions to 2D XYM - noData].

Parameters

<i>geom</i>	pointer to Geometry object [origin].
<i>no_data</i>	the default M value

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaCastGeomCollToXYM](#)

Note

the newly created object is an exact copy of the original one; except in that any elementary item will be cast to 2D [XYM] dimensions.

6.7.2.33 gaiaCastGeomCollToXYZ()

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaCastGeomCollToXYZ (
    gaiaGeomCollPtr geom )
```

Duplicates a Geometry object [casting dimensions to 3D XYZ].

Parameters

<i>geom</i>	pointer to Geometry object [origin].
-------------	--------------------------------------

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaCloneGeomColl](#), [gaiaCastGeomCollToXY](#), [gaiaCastGeomCollToXYM](#), [gaiaCastGeomCollToXYZM](#),
[gaiaCastGeomCollToXYZnoData](#)

Note

the newly created object is an exact copy of the original one; except in that any elementary item will be cast to 3D [XYZ] dimensions.

6.7.2.34 gaiaCastGeomCollToXYZM()

```
GAIA_GEO_DECLARE gaiaGeomCollPtr gaiaCastGeomCollToXYZM (
    gaiaGeomCollPtr geom )
```

Duplicates a Geometry object [casting dimensions to 3D XYZM].

Parameters

<i>geom</i>	pointer to Geometry object [origin].
-------------	--------------------------------------

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaCloneGeomColl](#), [gaiaCastGeomCollToXY](#), [gaiaCastGeomCollToXYZ](#), [gaiaCastGeomCollToXYM](#),
[gaiaCastGeomCollToXYZMnoData](#)

Note

the newly created object is an exact copy of the original one; except in that any elementary item will be cast to 3D [XYZM] dimensions.

6.7.2.35 [gaiaCastGeomCollToXYZMnoData\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaCastGeomCollToXYZMnoData (  
    gaiaGeomCollPtr geom,  
    double z_no_data,  
    double m_no_data )
```

Duplicates a Geometry object [casting dimensions to 3D XYZM - noData].

Parameters

<i>geom</i>	pointer to Geometry object [origin].
<i>z_no_data</i>	the default Z value
<i>m_no_data</i>	the default M value

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaCastGeomCollToXYZM](#)

Note

the newly created object is an exact copy of the original one; except in that any elementary item will be cast to 3D [XYZM] dimensions.

6.7.2.36 gaiaCastGeomCollToXYZnoData()

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaCastGeomCollToXYZnoData (
    gaiaGeomCollPtr geom,
    double no_data )
```

Duplicates a Geometry object [casting dimensions to 3D XYZ - noData].

Parameters

<i>geom</i>	pointer to Geometry object [origin].
<i>no_data</i>	the default Z value

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaCostGeomCollToXYZ](#)

Note

the newly created object is an exact copy of the original one; except in that any elementary item will be cast to 3D [XYZ] dimensions.

6.7.2.37 gaiaCheckClockwise()

```
GAIA GEO_DECLARE int gaiaCheckClockwise (
    gaiaGeomCollPtr geom )
```

Checks for Clockwise Geometry object.

Parameters

<i>geom</i>	pointer to Geometry object
-------------	----------------------------

Returns

0 if the Geometry is not clockwise: otherwise any other different value.

Note

a Clockwise Geometry contains no Polygons, or alternatively contains only Clockwise Polygons. A Clockwise Polygon has a Clockwise exterior ring and all interior rings are Counter-Clockwise.

6.7.2.38 `gaiaCheckCounterClockwise()`

```
GAIA GEO_DECLARE int gaiaCheckCounterClockwise (
    gaiaGeomCollPtr geom )
```

Checks for CounterClockwise Geometry object.

Parameters

<i>geom</i>	pointer to Geometry object
-------------	----------------------------

Returns

0 if the Geometry is not counter-clockwise: otherwise any other different value.

Note

a CounterClockwise Geometry contains no Polygons, or alternatively contains only CounterClockwise Polygons. A CounterClockwise Polygon has a CounterClockwise exterior ring and all interior rings are Clockwise.

6.7.2.39 `gaiaClockwise()`

```
GAIA GEO_DECLARE void gaiaClockwise (
    gaiaRingPtr p )
```

Determines the direction for a Ring object.

Parameters

<i>p</i>	pointer to Ring object
----------	------------------------

Returns

0 if the ring has counter-clockwise direction; any other different value for clockwise direction.

6.7.2.40 `gaiaCloneGeomColl()`

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaCloneGeomColl (
    gaiaGeomCollPtr geom )
```

Duplicates a Geometry object.

Parameters

<i>geom</i>	pointer to Geometry object [origin].
-------------	--------------------------------------

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaCloneLinestring](#), [gaiaCloneRing](#), [gaiaClonePolygon](#), [gaiaCloneGeomCollPoints](#), [gaiaCloneGeomCollLinestrings](#), [gaiaCloneGeomCollPolygons](#), [gaiaCastGeomCollToXY](#), [gaiaCastGeomCollToXYZ](#), [gaiaCastGeomCollToXYM](#), [gaiaCastGeomCollToXYZM](#), [gaiaExtractPointsFromGeomColl](#), [gaiaExtractLinestringsFromGeomColl](#), [gaiaExtractPolygonsFromGeomColl](#), [gaiaMergeGeometries](#), [gaiaCloneGeomCollSpecial](#)

Note

the newly created object is an exact copy of the original one.

6.7.2.41 [gaiaCloneGeomCollLinestrings\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaCloneGeomCollLinestrings (
    gaiaGeomCollPtr geom )
```

Duplicates a Geometry object [Linestrings only].

Parameters

<i>geom</i>	pointer to Geometry object [origin].
-------------	--------------------------------------

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaCloneLinestring](#), [gaiaCloneRing](#), [gaiaClonePolygon](#), [gaiaCloneGeomColl](#), [gaiaCloneGeomCollPoints](#), [gaiaCloneGeomCollPolygons](#)

Note

the newly created object is an exact copy of the original one; except in that only Linestrings objects will be copied.

Caveat: an empty Geometry could be returned.

6.7.2.42 [gaiaCloneGeomCollPoints\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaCloneGeomCollPoints (
    gaiaGeomCollPtr geom )
```

Duplicates a Geometry object [Points only].

Parameters

<i>geom</i>	pointer to Geometry object [origin].
-------------	--------------------------------------

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaCloneLinestring](#), [gaiaCloneRing](#), [gaiaClonePolygon](#), [gaiaCloneGeomColl](#), [gaiaCloneGeomCollLinestrings](#), [gaiaCloneGeomCollPolygons](#)

Note

the newly created object is an exact copy of the original one; except in that only Point objects will be copied.
Caveat: an empty Geometry could be returned.

6.7.2.43 `gaiaCloneGeomCollPolygons()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaCloneGeomCollPolygons (
    gaiaGeomCollPtr geom )
```

Duplicates a Geometry object [Polygons only].

Parameters

<i>geom</i>	pointer to Geometry object [origin].
-------------	--------------------------------------

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaCloneLinestring](#), [gaiaCloneRing](#), [gaiaClonePolygon](#), [gaiaCloneGeomColl](#), [gaiaCloneGeomCollPoints](#), [gaiaCloneGeomCollLinestrings](#)

Note

the newly created object is an exact copy of the original one; except in that only Polygons objects will be copied.
Caveat: an empty Geometry could be returned.

6.7.2.44 `gaiaCloneGeomCollSpecial()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaCloneGeomCollSpecial (
    gaiaGeomCollPtr geom,
    int mode )
```

Duplicates a Geometry object (special)

Parameters

<i>geom</i>	pointer to Geometry object [origin].
<i>mode</i>	one of GAIA_SAME_ORDER, GAIA_REVERSE_ORDER or GAIA_LHR_ORDER.

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

[gaiaCloneLinestringSpecial](#), [gaiaCloneRingSpecial](#), [gaiaClonePolygonSpecial](#), [gaiaCloneGeomColl](#)

Note

if GAIA_REVERSE_ORDER is specified, then any Linestring and/or Ring into the newly created object will be in reverse order. If GAIA_LHR_ORDER is specified instead, any Polygong will have the Exterior Ring in clockwise orientation, and any Interior Ring int counter-clockwise orientation. In any other case this function will simply default to `gaiaCloneGeomColl`.

6.7.2.45 `gaiaCloneLinestring()`

```
GAIAGEO_DECLARE gaiaLinestringPtr gaiaCloneLinestring (
    gaiaLinestringPtr line )
```

Duplicates a Linestring object.

Parameters

<i>line</i>	pointer to Linestring object [origin].
-------------	--

Returns

the pointer to newly created Linestring object: NULL on failure.

See also

[gaiaCloneRing](#), [gaiaClonePolygon](#), [gaiaCloneGeomColl](#), [gaiaCloneGeomCollPoints](#), [gaiaCloneGeomCollLinestrings](#), [gaiaCloneGeomCollPolygons](#), [gaiaCloneLinestringSpecial](#)

Note

the newly created object is an exact copy of the original one.

6.7.2.46 gaiaCloneLinestringSpecial()

```
GAIA_GEO_DECLARE gaiaLinestringPtr gaiaCloneLinestringSpecial (
    gaiaLinestringPtr line,
    int mode )
```

Duplicates a Linestring object (special)

Parameters

<i>line</i>	pointer to Linestring object [origin].
<i>mode</i>	one of GAIA_SAME_ORDER or GAIA_REVERSE_ORDER.

Returns

the pointer to newly created Linestring object: NULL on failure.

See also

[gaiaCloneLinestring](#), [gaiaCloneGeomCollSpecial](#)

Note

if GAIA_REVERSE_ORDER is specified, then any vertex into the newly created object will be in reverse order [first vertex will be last one, and last vertex will be the first one]. In any other case this function will simply default to [gaiaCloneLinestring](#).

6.7.2.47 gaiaClonePolygon()

```
GAIA_GEO_DECLARE gaiaPolygonPtr gaiaClonePolygon (
    gaiaPolygonPtr polyg )
```

Duplicates a Polygon object.

Parameters

<i>polyg</i>	pointer to Polygon object [origin].
--------------	-------------------------------------

Returns

the pointer to newly created Polygon object: NULL on failure.

See also

[gaiaCloneLinestring](#), [gaiaCloneRing](#), [gaiaCloneGeomColl](#), [gaiaCloneGeomCollPoints](#), [gaiaCloneGeomCollLinestrings](#), [gaiaCloneGeomCollPolygons](#), [gaiaClonePolygonSpecial](#)

Note

the newly created object is an exact copy of the original one.

6.7.2.48 [gaiaClonePolygonSpecial\(\)](#)

```
GAIAGEO_DECLARE gaiaPolygonPtr gaiaClonePolygonSpecial (  
    gaiaPolygonPtr polyg,  
    int mode )
```

Duplicates a Polygon object (special)

Parameters

<i>polyg</i>	pointer to Polygon object [origin].
<i>mode</i>	one of GAIA_SAME_ORDER, GAIA_REVERSE_ORDER, or GAIA_LHR_ORDER.

Returns

the pointer to newly created Polygon object: NULL on failure.

See also

[gaiaClonePolygon](#), [gaiaCloneGeomCollSpecial](#)

Note

if GAIA_REVERSE_ORDER is specified, then any Ring into the newly created object will be in reverse order. If GAIA_CW_ORDER is specified, any Exterior Ring will have clockwise orientation, and any Interior Ring will have counter-clockwise orientation. If GAIA_CCW_ORDER is specified, any Exterior Ring will have counter-clockwise orientation, and any Interior Ring will have clockwise orientation. In any other case this function will simply default to [gaiaClonePolygon](#).

6.7.2.49 [gaiaCloneRing\(\)](#)

```
GAIAGEO_DECLARE gaiaRingPtr gaiaCloneRing (  
    gaiaRingPtr ring )
```

Duplicates a Ring object.

Parameters

<i>ring</i>	pointer to Ring object [origin].
-------------	----------------------------------

Returns

the pointer to newly created Ring object: NULL on failure.

See also

[gaiaCloneLinestring](#), [gaiaClonePolygon](#), [gaiaCloneGeomColl](#), [gaiaCloneGeomCollPoints](#), [gaiaCloneGeomCollLinestrings](#), [gaiaCloneGeomCollPolygons](#), [gaiaCloneRingSpecial](#)

Note

the newly created object is an exact copy of the original one.

6.7.2.50 `gaiaCloneRingSpecial()`

```
GAIAGEO_DECLARE gaiaRingPtr gaiaCloneRingSpecial (
    gaiaRingPtr ring,
    int mode )
```

Duplicates a Ring object (special)

Parameters

<i>ring</i>	pointer to Ring object [origin].
<i>mode</i>	one of GAIA_SAME_ORDER or GAIA_REVERSE_ORDER.

Returns

the pointer to newly created Ring object: NULL on failure.

See also

[gaiaCloneRing](#), [gaiaClonePolygonSpecial](#)

Note

if GAIA_REVERSE_ORDER is specified, then any vertex into the newly created object will be in reverse order [first vertex will be last one, and last vertex will be the first one]. In any other case this function will simply default to [gaiaCloneRing](#).

6.7.2.51 `gaiaConvertLength()`

```
GAIA GEO_DECLARE int gaiaConvertLength (
    double value,
    int unit_from,
    int unit_to,
    double * cvt )
```

Convert a Length from a Measure Unit to another.

Parameters

<i>value</i>	the length measure to be converted.
<i>unit_from</i>	original Measure Unit.
<i>unit_to</i>	converted Measure Unit.
<i>cvt</i>	on completion this variable will contain the converted length measure.

Note

supported Measu Units are: GAIA_KM, GAIA_M, GAIA_DM, GAIA_CM, GAIA_MM, GAIA_KMI, GAIA_IN, GAIA_FT, GAIA_YD, GAIA_MI, GAIA_FATH, GAIC_CH, GAIA_LINK, GAIA_US_IN, GAIA_US_FT, GAIA_↔US_YD, GAIA_US_CH, GAIA_US_MI, GAIA_IND_YD, GAIA_IND_FT, GAIA_IND_CH

6.7.2.52 `gaiaCopyLinestringCoords()`

```
GAIA GEO_DECLARE void gaiaCopyLinestringCoords (
    gaiaLinestringPtr dst,
    gaiaLinestringPtr src )
```

Copies coordinates between two LINESTRING objects.

Parameters

<i>dst</i>	destination LINESTRING [output]
<i>src</i>	origin LINESTRING [input]

See also

[gaiaCopyLinestringCoordsReverse](#), [gaiaCopyLinestringCoordsEx](#)

Note

both LINESTRING objects must have exactly the same number of points: if dimensions aren't the same for both objects, then the appropriate conversion will be silently applied.

6.7.2.53 `gaiaCopyLinestringCoordsEx()`

```
GAIA GEO_DECLARE void gaiaCopyLinestringCoordsEx (
    gaiaLinestringPtr dst,
    gaiaLinestringPtr src,
    double z_no_data,
    double m_no_data )
```

Copies coordinates between two LINESTRING objects.

Parameters

<i>dst</i>	destination LINESTRING [output]
<i>src</i>	origin LINESTRING [input]
<i>z_no_data</i>	the default Z value \parma <i>m_no_data</i> the default M value

See also

[gaiaCopyLinestringCoords](#)

Note

both LINESTRING objects must have exactly the same number of points: if dimensions aren't the same for both objects, then the appropriate conversion will be silently applied.

6.7.2.54 `gaiaCopyLinestringCoordsReverse()`

```
GAIA GEO_DECLARE void gaiaCopyLinestringCoordsReverse (
    gaiaLinestringPtr dst,
    gaiaLinestringPtr src )
```

Copies coordinates between two LINESTRING objects in reverse order.

Parameters

<i>dst</i>	destination LINESTRING [output]
<i>src</i>	origin LINESTRING [input]

See also

[gaiaCopyLinestringCoords](#)

Note

both LINESTRING objects must have exactly the same number of points: if dimensions aren't the same for both objects, then the appropriate conversion will be silently applied.

6.7.2.55 `gaiaCopyRingCoords()`

```
GAIA GEO_DECLARE void gaiaCopyRingCoords (
    gaiaRingPtr dst,
    gaiaRingPtr src )
```

Copies coordinates between two RING objects.

Parameters

<i>dst</i>	destination RING [output]
<i>src</i>	origin RING [input]

See also

[gaiaCopyRingCoordsReverse](#), [gaiaCopyRingCoordEx](#)

Note

both RING objects must have exactly the same number of points: if dimensions aren't the same for both objects, then the appropriate conversion will be silently applied.

6.7.2.56 `gaiaCopyRingCoordsEx()`

```
GAIA GEO_DECLARE void gaiaCopyRingCoordsEx (
    gaiaRingPtr dst,
    gaiaRingPtr src,
    double z_no_data,
    double m_no_data )
```

Copies coordinates between two RING objects.

Parameters

<i>dst</i>	destination RING [output]
<i>src</i>	origin RING [input]
<i>z_no_data</i>	the default Z value
<i>m_no_data</i>	the default M value

See also

[gaiaCopyRingCoords](#)

Note

both RING objects must have exactly the same number of points: if dimensions aren't the same for both objects, then the appropriate conversion will be silently applied.

6.7.2.57 `gaiaCopyRingCoordsReverse()`

```
GAIA GEO_DECLARE void gaiaCopyRingCoordsReverse (
    gaiaRingPtr dst,
    gaiaRingPtr src )
```

Copies coordinates between two RING objects in reverse order.

Parameters

<i>dst</i>	destination RING [output]
<i>src</i>	origin RING [input]

See also

[gaiaCopyRingCoords](#)

Note

both RING objects must have exactly the same number of points: if dimensions aren't the same for both objects, then the appropriate conversion will be silently applied.

6.7.2.58 `gaiaCreatePolygon()`

```
GAIA GEO_DECLARE gaiaPolygonPtr gaiaCreatePolygon (
    gaiaRingPtr ring )
```

Allocates a POLYGON.

Parameters

<i>ring</i>	pointer to a valid RING object: assumed to be the Polygon's Exterior Ring.
-------------	--

Returns

the pointer to newly created POLYGON object: NULL on failure

See also

[gaiaAllocRing](#), [gaiaAllocRingXYZ](#), [gaiaAllocRingXYM](#), [gaiaAllocRingXYZM](#), [gaiaFreePolygon](#)

Note

you are responsible to destroy (before or after) any allocated POLYGON, unless you've passed ownership of the POLYGON object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

Ownership of passed Ring object will be transferred to the Polygon object being created.

6.7.2.59 `gaiaCurvosityIndex()`

```
GAIA GEO_DECLARE double gaiaCurvosityIndex (
    const void * p_cache,
    gaiaLinestringPtr line,
    int extra_points )
```

Computes the Curvosity Index for some Linestrings.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>line</i>	a generic Linestring.
<i>extra_points</i>	number of points to be interpolated at regular distance into the reference line.

Returns

the calculated Curvosity Index (expected to be in the range between 0.0 and 1.0).

6.7.2.60 `gaiaDimension()`

```
GAIA GEO_DECLARE int gaiaDimension (
    gaiaGeomCollPtr geom )
```

Determines OGC dimensions for a Geometry object.

Parameters

<i>geom</i>	pointer to Geometry object
-------------	----------------------------

Returns

OGC dimensions

Note

OGC dimensions are defined as follows:

- if the Geometry doesn't contain any elementary item: **-1**
- if the Geometry only contains Point items: **0**
- if the Geometry only contains Point / Linestring items: **1**
- if the Geometry contains some Polygon item: **2**

Examples

[demo2.c](#).

6.7.2.61 `gaiaDissolvePoints()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaDissolvePoints (
    gaiaGeomCollPtr geom )
```

Attempts to resolve a collection of Points from a Geometry object.

Parameters

<i>geom</i>	pointer to Geometry object.
-------------	-----------------------------

Returns

the pointer to newly created Geometry: NULL on failure.

See also

[gaiaLinearize](#), [gaiaDissolveSegments](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry created by [gaiaDissolvePoints\(\)](#)
the input Geometry can be of any arbitrary type:

- any Point will be copied untouched.
- any Linestring will be dissolved into sparse Points.
- any Ring will be dissolved into sparse Points.

6.7.2.62 `gaiaDissolveSegments()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaDissolveSegments (
    gaiaGeomCollPtr geom )
```

Attempts to resolve a collection of Segments from a Geometry object.

Parameters

<i>geom</i>	pointer to Geometry object.
-------------	-----------------------------

Returns

the pointer to newly created Geometry: NULL on failure.

See also

[gaiaLinearize](#), [gaiaDissolvePoints](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry created by [gaiaDissolveSegments\(\)](#)
the input Geometry can be of any arbitrary type:

- any Point will be copied untouched.
- any Linestring will be dissolved into Segments.
- any Ring will be dissolved into Segments.

6.7.2.63 gaiaEllipseParams()

```
GAIAGEO_DECLARE int gaiaEllipseParams (
    const char * name,
    double * a,
    double * b,
    double * rf )
```

Retrieves Geodesic params for an Ellipsoid definition.

Parameters

<i>name</i>	text string identifying an Ellipsoid definition.
<i>a</i>	on completion this variable will contain the first geodesic param.
<i>b</i>	on completion this variable will contain the second geodesic param.
<i>rf</i>	on completion this variable will contain the third geodesic param.

Returns

0 on failure: any other value on success.

See also

[gaiaGreatCircleDistance](#), [gaiaGeodesicDistance](#), [gaiaGreatCircleTotalLength](#), [gaiaGeodesicTotalLength](#)

Note

supported Ellipsoid definitions are: **MERIT**, **SGS85**, **GRS80**, **IAU76**, **airy**, **APL4.9**, **NWL9D**, **mod_airy**, **andreae**, **aust_SA**, **GRS67**, **bessel**, **bess_nam**, **clrk66**, **clrk80**, **CPM**, **delmbr**, **engelis**, **evrst30**, **evrst48**, **evrst56**, **evrst69**, **evrstSS**, **fschr60**

6.7.2.64 gaiaEnsureClosedRings()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaEnsureClosedRings (
    gaiaGeomCollPtr org )
```

Attempts to sanitize a possibly malformed Geometry object.

Parameters

<i>org</i>	pointer to Geometry object.
------------	-----------------------------

Returns

the pointer to newly created Geometry: NULL on failure.

See also

[gaiaIsToxic](#), [gaiaSanitize](#), [gaiaRemoveRepeatedPoint](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry created by [gaiaSanitize\(\)](#)

the output Geometry will surely have proper Ring closure: for sure any Ring will have exactly coinciding first and last Points.

6.7.2.65 [gaiaExtractLinestringsFromGeomColl\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaExtractLinestringsFromGeomColl (  
    gaiaGeomCollPtr geom )
```

Extracts any Linestring from a Geometry object.

Parameters

<i>geom</i>	pointer to Geometry object
-------------	----------------------------

Returns

the pointer to newly created Geometry: NULL on failure.

See also

[gaiaExtractPointsFromGeomColl](#), [gaiaExtractPolygonsFromGeomColl](#), [gaiaCloneGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry created by [gaiaExtractLinestringsFromGeomColl\(\)](#)

the newly created Geometry will contain any Linestring contained into the input Geometry.

if the input Geometry doesn't contains any Linestring, then NULL will be returned.

6.7.2.66 `gaiaExtractPointsFromGeomColl()`

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaExtractPointsFromGeomColl (
    gaiaGeomCollPtr geom )
```

Extracts any Point from a Geometry object.

Parameters

<i>geom</i>	pointer to Geometry object
-------------	----------------------------

Returns

the pointer to newly created Geometry: NULL on failure.

See also

[gaiaExtractLinestringsFromGeomColl](#), [gaiaExtractPolygonsFromGeomColl](#), [gaiaCloneGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry created by [gaiaExtractPointsFromGeomColl\(\)](#)
the newly created Geometry will contain any Point contained into the input Geometry.
if the input Geometry doesn't contains any Point, then NULL will be returned.

6.7.2.67 `gaiaExtractPolygonsFromGeomColl()`

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaExtractPolygonsFromGeomColl (
    gaiaGeomCollPtr geom )
```

Extracts any Polygon from a Geometry object.

Parameters

<i>geom</i>	pointer to Geometry object
-------------	----------------------------

Returns

the pointer to newly created Geometry: NULL on failure.

See also

[gaiaExtractPointsFromGeomColl](#), [gaiaExtractLinestringsFromGeomColl](#), [gaiaCloneGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry created by [gaiaExtractPolygonsFromGeomColl\(\)](#) the newly created Geometry will contain any Polygon contained into the input Geometry. if the input Geometry doesn't contains any Polygon, then NULL will be returned.

6.7.2.68 gaiaFree()

```
GAIGEO_DECLARE void gaiaFree (
    void * ptr )
```

Safely frees any dynamic memory block allocated by the library itself.

Parameters

<i>ptr</i>	pointer to dynamically allocated memory
------------	---

Note

on some platforms (most notably, Microsoft Windows) many different runtime libraries may actually support the same process. attempting to free() a memory block allocated by a different runtime module may easily cause fatal memory corruption.

6.7.2.69 gaiaFreeGeomColl()

```
GAIGEO_DECLARE void gaiaFreeGeomColl (
    gaiaGeomCollPtr geom )
```

Destroys a Geometry object.

Parameters

<i>geom</i>	pointer to the Geometry object to be destroyed
-------------	--

See also

[gaiaAllocGeomColl](#), [gaiaAllocGeomCollXYZ](#), [gaiaAllocGeomCollXYM](#), [gaiaAllocGeomCollXYZM](#)

Note

attempting to destroy any Geometry object whose ownership has already been transferred to some other (higher order) object is a serious error, and will easily cause severe memory corruption. Ownership of each POINT, LINESTRING or POLYGON object referenced by a Geometry object always belongs to the Geometry itself, so destroying the Geometry will surely destroy any related elementary geometry item as well.

Examples

[demo1.c](#), [demo2.c](#), [demo3.c](#), and [demo4.c](#).

6.7.2.70 `gaiaFreeLinestring()`

```
GAIAGEO_DECLARE void gaiaFreeLinestring (
    gaiaLinestringPtr ptr )
```

Destroys a LINESTRING object.

Parameters

<i>ptr</i>	pointer to the LINESTRING object to be destroyed
------------	--

See also

[gaiaAllocLinestring](#), [gaiaAllocLinestringXYZ](#), [gaiaAllocLinestringXYM](#), [gaiaAllocLinestringXYZM](#)

Note

attempting to destroy any LINESTRING object whose ownership has already been transferred to some other (higher order) object is a serious error, and will easily cause severe memory corruption.

6.7.2.71 `gaiaFreePoint()`

```
GAIAGEO_DECLARE void gaiaFreePoint (
    gaiaPointPtr ptr )
```

Destroys a POINT object.

Parameters

<i>ptr</i>	pointer to the POINT object to be destroyed
------------	---

See also

[gaiaAllocPoint](#), [gaiaAllocPointXYZ](#), [gaiaAllocPointXYM](#), [gaiaAllocPointXYZM](#)

Note

attempting to destroy any POINT object whose ownership has already been transferred to some other (higher order) object is a serious error, and will easily cause severe memory corruption.

6.7.2.72 `gaiaFreePolygon()`

```
GAIA GEO_DECLARE void gaiaFreePolygon (
    gaiaPolygonPtr polyg )
```

Destroys a POLYGON object.

Parameters

<i>polyg</i>	pointer to the POLYGON object to be destroyed
--------------	---

See also

[gaiaAllocPolygon](#), [gaiaAllocPolygonXYZ](#), [gaiaAllocPolygonXYM](#), [gaiaAllocPolygonXYZM](#), [gaiaCreatePolygon](#)

Note

attempting to destroy any POLYGON object whose ownership has already been transferred to some other (higher order) object is a serious error, and will easily cause severe memory corruption. Ownership of each RING object referenced by a POLYGON object always belongs to the POLYGON itself, so destroying the POLYGON will surely destroy any related RING as well.

6.7.2.73 `gaiaFreeRing()`

```
GAIA GEO_DECLARE void gaiaFreeRing (
    gaiaRingPtr ptr )
```

Destroys a RING object.

Parameters

<i>ptr</i>	pointer to the RING object to be destroyed
------------	--

See also

[gaiaAllocRing](#), [gaiaAllocRingXYZ](#), [gaiaAllocRingXYM](#), [gaiaAllocRingXYZM](#)

Note

attempting to destroy any RING object whose ownership has already been transferred to some other (higher order) object is a serious error, and will easily cause severe memory corruption.

6.7.2.74 `gaiaGeodesicArcLength()`

```

GAIAGEO_DECLARE int gaiaGeodesicArcLength (
    sqlite3 * sqlite,
    const void * cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2,
    int return_type,
    double * retval )

```

Computes several Geodesic values based on the Distance between two Geometries.

Parameters

<i>db_handle</i>	handle to the current DB connection.
<i>cache</i>	the same memory pointer passed to the corresponding call to spatialite_init_ex() and returned by spatialite_alloc_connection()
<i>geom1</i>	the first Geometry.
<i>geom2</i>	the second Geometry.
<i>return_type</i>	selects wich value has be computed. Must be one between: GAIA_GEODESIC_ARC_LENGTH_METERS, GAIA_GEODESIC_ARC_LENGTH_DEGREES, GAIA_GEODESIC_CHORD_LENGTH_METERS, GAIA_GEODESIC_CHORD_LENGTH_DEGREES, GAIA_GEODESIC_CENTRAL_ANGLE_DEGREES, GAIA_GEODESIC_CENTRAL_ANGLE_RADIANS, GAIA_GEODESIC_ARC_AREA_METERS or GAIA_GEODESIC_ARC_HEIGHT_METERS.
<i>retval</i>	on completion this variable will contain the computed value.

Returns

0 on failure: any other value on success.

See also

[gaiaGeodesicDistance](#), [gaiaGeodesicTotalLength](#)

Note

Both *geom1* and *geom2* must share the same SRID, that is expected to be of the Geographic type (longitudes and latitudes).

Requires to be supported by a recent version of PROJ ($\geq 4.9.0$).

If not supported by GEOS only two POINT Geometries will be accepted.

6.7.2.75 `gaiaGeodesicDistance()`

```

GAIAGEO_DECLARE double gaiaGeodesicDistance (
    double a,
    double b,
    double rf,
    double lat1,
    double lon1,
    double lat2,
    double lon2 )

```

Calculates the Geodesic Distance between two Points.

Parameters

<i>a</i>	first geodesic parameter.
<i>b</i>	second geodesic parameter.
<i>rf</i>	third geodesic parameter.
<i>lat1</i>	Latitude of first Point.
<i>lon1</i>	Longitude of first Point.
<i>lat2</i>	Latitude of second Point.
<i>lon2</i>	Longitude of second Point.

Returns

the calculated Geodesic Distance.

See also

[gaiaEllipseParams](#), [gaiaGreatCircleDistance](#), [gaiaGreatCircleTotalLength](#), [gaiaGeodesicTotalLength](#), [gaiaGeodesicArcLength](#)

Note

the returned distance is expressed in Kilometers.
the Geodesic method is much more accurate but slowest to be calculated.

6.7.2.76 gaiaGeodesicTotalLength()

```
GAIA GEO_DECLARE double gaiaGeodesicTotalLength (
    double a,
    double b,
    double rf,
    int dims,
    double * coords,
    int vert )
```

Calculates the Geodesic Total Length for a Linestring / Ring.

Parameters

<i>a</i>	first geodesic parameter.
<i>b</i>	second geodesic parameter.
<i>rf</i>	third geodesic parameter.
<i>dims</i>	dimensions: one of GAIA_XY, GAIA_XY_Z, GAIA_XY_M or GAIA_XY_ZM
<i>coords</i>	pointed to COORD mem-array
<i>vert</i>	number of Points (aka Vertices) within the COORD mem-array

Returns

the calculated Geodesic Total Length.

See also

[gaiaEllipseParams](#), [gaiaGreatCircleDistance](#), [gaiaGeodesicDistance](#), [gaiaGreatCircleTotalLength](#), [gaiaGeodesicArcLength](#)

Note

the returned length is expressed in Kilometers.

the Geodesic method is much more accurate but slowest to be calculated.

dims, **coords** and **vert** are usually expected to correspond to **DimensionModel**, **Coords** and **Points** members from a [gaiaLinestringStruct](#) or [gaiaRingStruct](#).

6.7.2.77 [gaiaGeometryAliasType\(\)](#)

```
GAIA_GEO_DECLARE int gaiaGeometryAliasType (
    gaiaGeomCollPtr geom )
```

Determines the corresponding Type for a Geometry object.

Parameters

<i>geom</i>	pointer to Geometry object
-------------	----------------------------

Returns

the corresponding Geometry Type

See also

[gaiaGeometryType](#)

Note

Type is one of: GAIA_POINT, GAIA_LINESTRING, GAIA_POLYGON, GAIA_MULTIPPOINT, GAIA_↔MULTILINESTRING, GAIA_MULTIPOLYGON, GAIA_GEOMETRYCOLLECTION
on failure GAIA_NONE will be returned.

Remarks

deprecated function (used in earlier SpatiaLite versions).

6.7.2.78 [gaiaGeometryType\(\)](#)

```
GAIA_GEO_DECLARE int gaiaGeometryType (
    gaiaGeomCollPtr geom )
```

Determines the corresponding Type for a Geometry object.

Parameters

<i>geom</i>	pointer to Geometry object
-------------	----------------------------

Returns

the corresponding Geometry Type

Note

Type is one of: GAIA_POINT, GAIA_LINestring, GAIA_POLYGON, GAIA_MULTIPPOINT, GAIA_↵
MULTILINestring, GAIA_MULTIPOLYGON, GAIA_GEOMETRYCOLLECTION, GAIA_POINTZ, GAIA_↵
_LINestringZ, GAIA_POLYGONZ, GAIA_MULTIPPOINTZ, GAIA_MULTILINestringZ, GAIA_↵
MULTIPOLYGONZ, GAIA_GEOMETRYCOLLECTIONZ, GAIA_POINTM, GAIA_LINestringM, GAIA_↵
_POLYGONM, GAIA_MULTIPPOINTM, GAIA_MULTILINestringM, GAIA_MULTIPOLYGONM, GAIA_↵
GEOMETRYCOLLECTIONM, GAIA_POINTZM, GAIA_LINestringZM, GAIA_POLYGONZM, GAIA_↵
MULTIPPOINTZM, GAIA_MULTILINestringZM, GAIA_MULTIPOLYGONZM, GAIA_GEOMETRYCOLLECTIONZM

on failure GAIA_NONE will be returned.

Examples

[demo1.c](#), and [demo2.c](#).

6.7.2.79 gaiaGreatCircleDistance()

```
GAIA GEO_DECLARE double gaiaGreatCircleDistance (
    double a,
    double b,
    double lat1,
    double lon1,
    double lat2,
    double lon2 )
```

Calculates the Great Circle Distance between between two Points.

Parameters

<i>a</i>	first geodesic parameter.
<i>b</i>	second geodesic parameter.
<i>lat1</i>	Latitude of first Point.
<i>lon1</i>	Longitude of first Point.
<i>lat2</i>	Latitude of second Point.
<i>lon2</i>	Longitude of second Point.

Returns

the calculated Great Circle Distance.

See also

[gaiaEllipseParams](#), [gaiaGeodesicDistance](#), [gaiaGreatCircleTotalLength](#), [gaiaGeodesicTotalLength](#)

Note

the returned distance is expressed in Kilometers.
the Great Circle method is less accurate but fastest to be calculated.

6.7.2.80 `gaiaGreatCircleTotalLength()`

```
GAIA GEO_DECLARE double gaiaGreatCircleTotalLength (
    double a,
    double b,
    int dims,
    double * coords,
    int vert )
```

Calculates the Great Circle Total Length for a Linestring / Ring.

Parameters

<i>a</i>	first geodesic parameter.
<i>b</i>	second geodesic parameter.
<i>dims</i>	dimensions: one of GAIA_XY, GAIA_XY_Z, GAIA_XY_M or GAIA_XY_ZM
<i>coords</i>	pointed to COORD mem-array
<i>vert</i>	number of Points (aka Vertices) within the COORD mem-array

Returns

the calculated Great Circle Total Length.

See also

[gaiaEllipseParams](#), [gaiaGreatCircleDistance](#), [gaiaGeodesicDistance](#), [gaiaGeodesicTotalLength](#)

Note

the returned length is expressed in Kilometers.
the Great Circle method is less accurate but fastest to be calculated.
dims, **coords** and **vert** are usually expected to correspond to **DimensionModel**, **Coords** and **Points** members from a [gaiaLinestringStruct](#) or [gaiaRingStruct](#)

6.7.2.81 `gaiaInsertInteriorRing()`

```
GAIA GEO_DECLARE void gaiaInsertInteriorRing (
    gaiaPolygonPtr p,
    gaiaRingPtr ring )
```

Inserts an already existing Ring object into a Polygon object.

Parameters

<i>p</i>	pointer to the Polygon object
<i>ring</i>	pointer to the Ring object

See also

gaiaAddRingToPolygon

Note

ownership of the Ring object still remains to the calling procedure (a duplicated copy of the original Ring will be inserted into the Polygon).
 the newly created Polygon will have the same dimensions as the Ring has.
 if required the Polygon's Interior Rings count could be increased.

6.7.2.82 gaiaInsertLinestringInGeomColl()

```
GAIA GEO_DECLARE void gaiaInsertLinestringInGeomColl (
    gaiaGeomCollPtr p,
    gaiaLinestringPtr line )
```

Inserts an already existing Linestring object into a Geometry object.

Parameters

<i>p</i>	pointer to the Geometry object.
<i>line</i>	pointer to the Linestring object.

Note

ownership of the Linestring object will be transferred to the Geometry object.

6.7.2.83 gaiaInsertPolygonInGeomColl()

```
GAIA GEO_DECLARE gaiaPolygonPtr gaiaInsertPolygonInGeomColl (
    gaiaGeomCollPtr p,
    gaiaRingPtr ring )
```

Creates a new Polygon object into a Geometry object starting from an already existing Ring object.

Parameters

<i>p</i>	pointer to the Geometry object.
<i>ring</i>	pointer to the Ring object [assumed to represent to Polygon's Exterior Ring].

Returns

the pointer to the newly created Polygon object: NULL on failure.

Note

ownership of the Ring object will be transferred to the Polygon object, and the Polygon object ownerships belongs to the Geometry object.
the Polygon object will have the same dimensions as the Ring object has.

6.7.2.84 gaiaInterpolatePoint()

```
GAIA GEO_DECLARE int gaiaInterpolatePoint (
    const void * p_cache,
    gaiaGeomCollPtr line,
    gaiaGeomCollPtr point,
    double * m_value )
```

Will interpolate the M-value for a LinestringM at the point closest to the given Point.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>line</i>	pointer to Geometry object of the Linestring type and supporting the M dimension.
<i>point</i>	pointer to Geometry object of the Point type.
<i>m_value</i>	on succesfull completion this variable will contain the interpolated M value

Returns

0 on failure: any other value on success.

6.7.2.85 gaiaIntersect()

```
GAIA GEO_DECLARE int gaiaIntersect (
    double * x0,
    double * y0,
    double x1,
    double y1,
    double x2,
    double y2,
    double x3,
    double y3,
    double x4,
    double y4 )
```

Determines the intesection Point between two Segments.

Parameters

<i>x0</i>	on completion this variable will contain the Intersection X coord
<i>y0</i>	on completion this variable will contain the Intersection Y coord
<i>x1</i>	start Point X of first Segment
<i>y1</i>	start Point Y of first Segment
<i>x2</i>	end Point X of first Segment
<i>y2</i>	end Point Y of first Segment
<i>x3</i>	start Point X of second Segment
<i>y3</i>	start Point Y of second Segment
<i>x4</i>	end Point X of second Segment
<i>y4</i>	end Point Y of second Segment

Returns

0 if the Segments doesn't intersect at all: any other value on success.

6.7.2.86 gaialsEmpty()

```
GAIA GEO_DECLARE int gaiaIsEmpty (
    gaiaGeomCollPtr geom )
```

Checks for empty Geometry object.

Parameters

<i>geom</i>	pointer to Geometry object
-------------	----------------------------

Returns

0 if the Geometry is not empty: otherwise any other different value.

Note

an empty Geometry is a Geometry not containing any elementary item: i.e. no Points, no Linestrings and no Polygons at all.

6.7.2.87 gaialsNotClosedGeomColl()

```
GAIA GEO_DECLARE int gaiaIsNotClosedGeomColl (
    gaiaGeomCollPtr geom )
```

Checks for not-closed Rings in a Geometry object.

Parameters

<i>geom</i>	pointer to Geometry object
-------------	----------------------------

Returns

0 if the Geometry has no unclosed Rings: otherwise any other different value.

See also

[gaialsNotClosedGeomColl_r](#), [gaialsToxic](#), [gaialsNotClosedRing](#)

Note

This function allows to explicitly identify any Geometry containing at least one unclosed Ring.
not reentrant and thread unsafe.

6.7.2.88 gaialsNotClosedGeomColl_r()

```
GAIAGEO_DECLARE int gaiaIsNotClosedGeomColl_r (  
    const void * p_data,  
    gaiaGeomCollPtr geom )
```

Checks for not-closed Rings in a Geometry object.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to Geometry object

Returns

0 if the Geometry has no unclosed Rings: otherwise any other different value.

See also

[gaialsNotClosedGeomColl](#), [gaialsToxic](#), [gaialsNotClosedRing](#)

Note

This function allows to explicitly identify any Geometry containing at least one unclosed Ring.
reentrant and thread-safe.

6.7.2.89 gaialsNotClosedRing()

```
GAIA GEO_DECLARE int gaiaIsNotClosedRing (
    gaiaRingPtr ring )
```

Checks for not-closed Rings.

Parameters

<i>ring</i>	pointer to Ring object
-------------	------------------------

Returns

0 if the Ring is unclosed: otherwise any other different value.

See also

[gaialsNotClosedRing_r](#), [gaialsToxic](#), [gaialsNotClosedGeomColl](#)

Note

unclosed Rings cause GEOS supported functions to crash.
Spatialite will always carefully check any Ring before passing it to GEOS, eventually silently inserting a further point required so to properly close the figure.
This function allows to explicitly identify any unclosed Ring.
not reentrant and thread unsafe.

6.7.2.90 gaialsNotClosedRing_r()

```
GAIAGEO_DECLARE int gaiaIsNotClosedRing_r (  
    const void * p_data,  
    gaiaRingPtr ring )
```

Checks for not-closed Rings.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>ring</i>	pointer to Ring object

Returns

0 if the Ring is unclosed: otherwise any other different value.

See also

[gaialsNotClosedRing](#), [gaialsToxic](#), [gaialsNotClosedGeomColl](#)

Note

unclosed Rings cause GEOS supported functions to crash.
Spatialite will always carefully check any Ring before passing it to GEOS, eventually silently inserting a further point required so to properly close the figure.
This function allows to explicitly identify any unclosed Ring.
reentrant and thread-safe.

6.7.2.91 gaialsPointOnPolygonSurface()

```
GAIAGEO_DECLARE int gaiaIsPointOnPolygonSurface (
    gaiaPolygonPtr polyg,
    double x,
    double y )
```

Checks if a Point lays on a Polygon surface.

Parameters

<i>polyg</i>	pointer to Polygon object
<i>x</i>	Point X coordinate
<i>y</i>	Point Y coordinate

Returns

0 if false: any other value if true

6.7.2.92 gaialsPointOnRingSurface()

```
GAIAGEO_DECLARE int gaiaIsPointOnRingSurface (
    gaiaRingPtr ring,
    double pt_x,
    double pt_y )
```

Check if a Point lays on a Ring surface.

Parameters

<i>ring</i>	pointer to Ring object
<i>pt_x</i>	Point X coordinate
<i>pt_y</i>	Point Y coordinate

Returns

0 if false: any other value if true

6.7.2.93 gaialsToxic()

```
GAIAGEO_DECLARE int gaiaIsToxic (
    gaiaGeomCollPtr geom )
```

Checks for toxic Geometry object.

Parameters

<i>geom</i>	pointer to Geometry object
-------------	----------------------------

Returns

0 if the Geometry is not toxic: otherwise any other different value.

See also

[gaiaIsToxic_r](#), [gaiaSanitize](#)

Note

a **toxic** Geometry is a Geometry containing severely malformed Polygons: i.e. containing less than 4 Points. Or containing severely malformed Linestrings: i.e. containing less than 2 Points. Attempting to pass any toxic Geometry to GEOS supported functions will easily cause a crash. not reentrant and thread unsafe.

6.7.2.94 [gaiaIsToxic_r\(\)](#)

```
GAIAGEO_DECLARE int gaiaIsToxic_r (
    const void * p_cache,
    gaiaGeomCollPtr geom )
```

Checks for toxic Geometry object.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom</i>	pointer to Geometry object

Returns

0 if the Geometry is not toxic: otherwise any other different value.

See also

[gaiaIsToxic](#), [gaiaSanitize](#)

Note

a **toxic** Geometry is a Geometry containing severely malformed Polygons: i.e. containing less than 4 Points. Or containing severely malformed Linestrings: i.e. containing less than 2 Points. Attempting to pass any toxic Geometry to GEOS supported functions will easily cause a crash. reentrant and thread-safe.

6.7.2.95 `gaiaIsValidTrajectory()`

```
GAIAGEO_DECLARE int gaiaIsValidTrajectory (
    gaiaGeomCollPtr geom )
```

Checks if a Geometry object is valid Trajectory.

Parameters

<i>geom</i>	pointer to Geometry object
-------------	----------------------------

Returns

0 if false; any other value if true

See also

[gaiaTrajectoryInterpolatePoint](#)

Note

a Geometry is considered to be a valid Trajectory if it contains a simple LINESTRING supporting M-values growing from each vertex to the next.

6.7.2.96 `gaiaLinearize()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLinearize (
    gaiaGeomCollPtr geom,
    int force_multi )
```

Attempts to resolve a (Multi)Linestring from a Geometry object.

Parameters

<i>geom</i>	pointer to Geometry object.
<i>force_multi</i>	0 if the returned Geometry could represent a Linestring: any other value if casting to MultiLinestring is required unconditionally.

Returns

the pointer to newly created Geometry: NULL on failure.

See also

[gaiaDissolveSegments](#), [gaiaDissolvePoints](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry created by [gaiaLinearize\(\)](#)
the input Geometry is expected to contain Polygons only: then any Ring will be transformed into the corresponding Linestring.

6.7.2.97 gaiaLineGetPoint()

```
GAIA GEO_DECLARE int gaiaLineGetPoint (
    gaiaLinestringPtr ln,
    int v,
    double * x,
    double * y,
    double * z,
    double * m )
```

Gets coordinates from a Linestring's Point.

Parameters

<i>ln</i>	pointer to Linestring object.
<i>v</i>	relative position of Point: first Point has index 0
<i>x</i>	on completion this variable will contain the Point X coordinate.
<i>y</i>	on completion this variable will contain the Point Y coordinate.
<i>z</i>	on completion this variable will contain the Point Z coordinate.
<i>m</i>	on completion this variable will contain the Point M measure.

Returns

0 on failure: any other different value on success.

See also

[gaiaLineSetPoint](#), [gaiaGetPoint](#), [gaiaGetPointXYZ](#), [gaiaGetPointXYM](#), [gaiaGetPointXYZM](#)

Note

this function perform the same identical task performed by [gaiaGetPoint\(\)](#), [gaiaGetPointXYZ\(\)](#), [gaiaGetPointXYM\(\)](#) and [gaiaGetPointXYZM\(\)](#) macros.
using the [gaiaLineGetPoint\(\)](#) function is a little bit slower but is intrinsically safest, because misused macros can easily cause severe memory corruption.
[gaiaLineGetPoint\(\)](#) instead will always ensure that the appropriate dimensions (as declared by the Linestring object) will be correctly used.

6.7.2.98 `gaiaLineSetPoint()`

```
GAIA GEO_DECLARE int gaiaLineSetPoint (
    gaiaLinestringPtr ln,
    int v,
    double x,
    double y,
    double z,
    double m )
```

Sets coordinates for a Linestring's Point.

Parameters

<i>ln</i>	pointer to Linestring object.
<i>v</i>	relative position of Point: first Point has index 0
<i>x</i>	the Point's X coordinate.
<i>y</i>	the Point's Y coordinate.
<i>z</i>	the Point's Z coordinate.
<i>m</i>	the Point's M measure.

Returns

0 on failure: any other different value on success.

See also

[gaiaLineGetPoint](#), [gaiaSetPoint](#), [gaiaSetPointXYZ](#), [gaiaSetPointXYM](#), [gaiaSetPointXYZM](#)

Note

this function perform the same identical task performed by [gaiaSetPoint\(\)](#), [gaiaSetPointXYZ\(\)](#), [gaiaSetPointXYM\(\)](#) and [gaiaSetPointXYZM\(\)](#) macros.

using the [gaiaLineSetPoint\(\)](#) function is a little bit slower but is intrinsically safest, because misused macros can easily cause severe memory corruption.

[gaiaLineSetPoint\(\)](#) instead will always ensure that the appropriate dimensions (as declared by the Linestring object) will be correctly used.

6.7.2.99 `gaiaLinestringEquals()`

```
GAIA GEO_DECLARE int gaiaLinestringEquals (
    gaiaLinestringPtr line1,
    gaiaLinestringPtr line2 )
```

Checks if two Linestring objects are equivalent.

Parameters

<i>line1</i>	pointer to first Linestring object.
<i>line2</i>	pointer to second Linestring object.

Returns

0 if false: any other different value if true

See also

[gaiaPolygonEquals](#)

Note

two Linestrings objects are assumed to be equivalent if exactly

Remarks

deprecated function (used in earlier SpatiaLite versions). the same Points are found in both them.

6.7.2.100 gaiaLocateBetweenMeasures()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLocateBetweenMeasures (
    gaiaGeomCollPtr geom,
    double m_start,
    double m_end )
```

Return a GeometryCollection containing elements matching the specified range of measures.

Parameters

<i>geom</i>	pointer to Geometry object
<i>m_start</i>	range of measures: start value
<i>m_end</i>	range of measures: end value

Returns

the pointer to newly created Geometry: NULL on failure.

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry created by [gaiaLocateBetweenMeasures\(\)](#)
the newly created Geometry will contain Points and/or Linestrings.
if the input Geometry has no M dimension then NULL will be returned.
if the input Geometry doesn't contains any point/vertex corresponding to the required range of measures then NULL will be returned.
if the input Geometry contains any Polygon (or is a GeometryCollection) then NULL will be returned.

6.7.2.101 `gaiaMakeArc()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaMakeArc (
    double center_x,
    double center_y,
    double radius,
    double start,
    double stop,
    double step )
```

Creates a Circular Arc (Linestring) Geometry.

Parameters

<i>center↔ _x</i>	center point X coordinate.
<i>center↔ _y</i>	center point Y coordinate.
<i>radius</i>	the circle's radius.
<i>start</i>	the start angle (in degrees).
<i>start</i>	the stop angle (in degrees).
<i>step</i>	angular distance (in degrees) between points on the circumference.

See also

[gaiaMakeCircle](#), [gaiaMakeEllipse](#), [gaiaMakeEllipticArc](#)

Note

simply a convenience method defaulting to [gaiaMakeEllipticArc](#) with both axes set to radius value

6.7.2.102 `gaiaMakeCircle()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaMakeCircle (
    double center_x,
    double center_y,
    double radius,
    double step )
```

Creates a Circle (Linestring) Geometry.

Parameters

<i>center↔ _x</i>	center point X coordinate.
<i>center↔ _y</i>	center point Y coordinate.
<i>radius</i>	the circle's radius.
<i>step</i>	angular distance (in degrees) between points on the circumference.

See also

[gaiaMakeArc](#), [gaiaMakeEllipse](#), [gaiaMakeEllipticArc](#)

Note

simply a convenience method defaulting to [gaiaMakeEllipse](#) with both axes set to radius value

6.7.2.103 [gaiaMakeEllipse\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaMakeEllipse (
    double center_x,
    double center_y,
    double x_axis,
    double y_axis,
    double step )
```

Creates an Ellipse (Linestring) Geometry.

Parameters

<i>center_x</i>	center point X coordinate.
<i>center_y</i>	center point Y coordinate.
<i>x_axis</i>	the ellipses's X axis.
<i>y_axis</i>	the ellipses's Y axis.
<i>step</i>	angular distance (in degrees) between points on the ellipse.

See also

[gaiaMakeEllipticArc](#), [gaiaMakeCircle](#), [gaiaMakeArc](#)

6.7.2.104 [gaiaMakeEllipticArc\(\)](#)

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaMakeEllipticArc (
    double center_x,
    double center_y,
    double x_axis,
    double y_axis,
    double start,
    double stop,
    double step )
```

Creates an Elliptic Arc (Linestring) Geometry.

Parameters

<i>center</i> ↔ <i>_x</i>	center point X coordinate.
<i>center</i> ↔ <i>_y</i>	center point Y coordinate.
<i>x_axis</i>	the ellipses's X axis.
<i>y_axis</i>	the ellipses's Y axis.
<i>start</i>	the start angle (in degrees).
<i>stop</i>	the stop angle (in degrees).
<i>step</i>	angular distance (in degrees) between points on the ellipse.

See also

[gaiaMakeCircle](#), [gaiaMakeEllipse](#), [gaiaMakeEllipticArc](#)

6.7.2.105 `gaiaMakePolygon()`

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaMakePolygon (
    gaiaGeomCollPtr exterior,
    gaiaGeomCollPtr interiors )
```

Creates a Polygon from closed Linestrings.

Parameters

<i>exterior</i>	a closed Linestring assumed to represent the Exterior Ring.
<i>interiors</i>	one (or more than one) closed Linestrings assumed to represent all Interior Rings (could be a Linestring or a MultiLinestring). NULL if there are no Interior Rings at all.

See also

[gaiaPolygonize](#)

Note

this method will simply check if all the received Linestrings are closed, but it could possibly return an invalid Polygon if there is any topology inconsistency between the exterior and interior rings. You are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaPolygonize\(\)](#) not reentrant and thread unsafe.

6.7.2.106 `gaiaMeasureArea()`

```
GAIA GEO_DECLARE double gaiaMeasureArea (
    gaiaRingPtr ring )
```

Measures the geometric area for a Ring object.

Parameters

<i>ring</i>	pointer to Ring object
-------------	------------------------

Returns

the calculated geometric area

See also

[gaiaGeomCollArea](#)

Remarks

internal method: doesn't require any GEOS support.

6.7.2.107 `gaiaMeasureLength()`

```
GAIA GEO_DECLARE double gaiaMeasureLength (
    int dims,
    double * coords,
    int vert )
```

Measures the geometric length for a Linestring or Ring.

Parameters

<i>dims</i>	dimensions: one of GAIA_XY, GAIA_XY_Z, GAIA_XY_M or GAIA_XY_ZM
<i>coords</i>	pointed to COORD mem-array
<i>vert</i>	number of Points (aka Vertices) within the COORD mem-array

Returns

the calculated geometric length

See also

[gaiaGeomCollLength](#)

Note

dims, **coords** and **vert** are usually expected to correspond to **DimensionModel**, **Coords** and **Points** members from a [gaiaLinestringStruct](#) or [gaiaRingStruct](#)

Remarks

internal method: doesn't require any GEOS support.

6.7.2.108 `gaiaMergeGeometries()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaMergeGeometries (
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Merges two Geometry objects into a single one.

Parameters

<i>geom1</i>	pointer to first Geometry object.
<i>geom2</i>	pointer to second Geometry object.

Returns

the pointer to newly created Geometry: NULL on failure.

See also

[gaiaMergeGeometries_r](#), [gaiaCloneGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry created by [gaiaMergeGeometries\(\)](#)
the newly created Geometry will contain any Point, Linestring and/or Polygon contained in both input Geometries.
not reentrant and thread unsafe.

6.7.2.109 `gaiaMergeGeometries_r()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaMergeGeometries_r (
    const void * p_cache,
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2 )
```

Merges two Geometry objects into a single one.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>geom1</i>	pointer to first Geometry object.
<i>geom2</i>	pointer to second Geometry object.

Returns

the pointer to newly created Geometry: NULL on failure.

See also

[gaiaMergeGeometries](#), [gaiaCloneGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry created by [gaiaMergeGeometries\(\)](#) the newly created Geometry will contain any Point, Linestring and/or Polygon contained in both input Geometries.
reentrant and thread-safe.

6.7.2.110 [gaiaMinDistance\(\)](#)

```
GAIA GEO_DECLARE double gaiaMinDistance (
    double x0,
    double y0,
    int dims,
    double * coords,
    int vert )
```

Computes the minimum distance between a Point and a Linestring or Ring.

Parameters

<i>x0</i>	Point X coordinate
<i>y0</i>	Point Y coordinate
<i>dims</i>	dimensions: one of GAIA_XY, GAIA_XY_Z, GAIA_XY_M or GAIA_XY_ZM
<i>coords</i>	pointed to COORD mem-array
<i>vert</i>	number of Points (aka Vertices) within the COORD mem-array

Returns

the calculated minimum distance.

Note

dims, **coords** and **vert** are usually expected to correspond to **DimensionModel**, **Coords** and **Points** members from a [gaiaLinestringStruct](#) or [gaiaRingStruct](#)

6.7.2.111 [gaiaNormalizeLonLat\(\)](#)

```
GAIA GEO_DECLARE void gaiaNormalizeLonLat (
    gaiaGeomCollPtr geom )
```

Shifts any coordinate to within the "normal range" of longitude and latitude values (-180.0 to 180.0 longitude and -90.0 to 90.0 latitude).

Parameters

<i>geom</i>	pointer to Geometry object.
-------------	-----------------------------

See also

[gaiaScaleCoords](#), [gaiaRotateCoords](#), [gaiaReflectCoords](#), [gaiaSwapCoords](#), [gaiaShiftCoords3D](#), [gaiaShiftLongitude](#)

6.7.2.112 `gaiaPolygonEquals()`

```
GAIAGEO_DECLARE int gaiaPolygonEquals (
    gaiaPolygonPtr polyg1,
    gaiaPolygonPtr polyg2 )
```

Checks if two Polygons objects are equivalent.

Parameters

<i>polyg1</i>	pointer to first Polygon object.
<i>polyg2</i>	pointer to second Polygon object.

Returns

0 if false: any other different value if true

See also

[gaiaLinestringEquals](#)

Note

two Polygon objects are assumed to be equivalent if exactly the same Points are found in both them.

Remarks

deprecated function (used in earlier SpatiaLite versions).

6.7.2.113 `gaiaReflectCoords()`

```
GAIAGEO_DECLARE void gaiaReflectCoords (
    gaiaGeomCollPtr geom,
    int x_axis,
    int y_axis )
```

Reflects any coordinate within a Geometry object.

Parameters

<i>geom</i>	pointer to Geometry object.
<i>x_axis</i>	if set to 0, no X axis reflection will be applied: otherwise the X axis will be reflected.
<i>y_axis</i>	if set to 0, no Y axis reflection will be applied: otherwise the Y axis will be reflected.

See also

[gaiaShiftCoords](#), [gaiaScaleCoords](#), [gaiaRotateCoords](#), [gaiaSwapCoords](#)

6.7.2.114 `gaiaRemoveRepeatedPoints()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaRemoveRepeatedPoints (
    gaiaGeomCollPtr org,
    double tolerance )
```

Attempts to sanitize a possibly malformed Geometry object.

Parameters

<i>org</i>	pointer to Geometry object.
<i>tolerance</i>	

Returns

the pointer to newly created Geometry: NULL on failure.

See also

[gaiaIsToxic](#), [gaiaSanitizeGeometry](#), [gaiaEnsureClosedRings](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry created by [gaiaSanitize\(\)](#)
the output Geometry will surely have no repeated Points on Linestrings or Rings or MultiPoints (i.e. consecutive Points sharing exactly the same coordinates or falling within the given tolerance): any repeated Point will be suppressed, simply leaving only the first occurrence.

6.7.2.115 `gaiaRingCentroid()`

```
GAIAGEO_DECLARE void gaiaRingCentroid (
    gaiaRingPtr ring,
    double * rx,
    double * ry )
```

Determines the Centroid for a Ring object.

Parameters

<i>ring</i>	pointer to Ring object.
<i>rx</i>	on completion this variable will contain the centroid X coordinate.
<i>ry</i>	on completion this variable will contain the centroid Y coordinate.

See also

[gaiaGeomCollCentroid](#)

Remarks

internal method: doesn't require any GEOS support.

6.7.2.116 gaiaRingGetPoint()

```
GAIA GEO_DECLARE int gaiaRingGetPoint (
    gaiaRingPtr rng,
    int v,
    double * x,
    double * y,
    double * z,
    double * m )
```

Gets coordinates from a Ring's Point.

Parameters

<i>rng</i>	pointer to Ring object.
<i>v</i>	relative position of Point: first Point has index 0
<i>x</i>	on completion this variable will contain the Point X coordinate.
<i>y</i>	on completion this variable will contain the Point Y coordinate.
<i>z</i>	on completion this variable will contain the Point Z coordinate.
<i>m</i>	on completion this variable will contain the Point M measure.

Returns

0 on failure: any other different value on success.

See also

[gaiaRingSetPoint](#), [gaiaGetPoint](#), [gaiaGetPointXYZ](#), [gaiaGetPointXYM](#), [gaiaGetPointXYZM](#)

Note

this function perform the same identical task performed by [gaiaGetPoint\(\)](#), [gaiaGetPointXYZ\(\)](#), [gaiaGetPointXYM\(\)](#) and [gaiaGetPointXYZM\(\)](#) macros.
 using the [gaiaRingGetPoint\(\)](#) function is a little bit slower but is intrinsically safest, because misused macros can easily cause severe memory corruption.
[gaiaRingGetPoint\(\)](#) instead will always ensure that the appropriate dimensions (as declared by the Ring object) will be correctly used.

6.7.2.117 gaiaRingSetPoint()

```
GAIA GEO_DECLARE int gaiaRingSetPoint (
    gaiaRingPtr rng,
    int v,
    double x,
    double y,
    double z,
    double m )
```

Sets coodinates for a Ring's Point.

Parameters

<i>rng</i>	pointer to Ring object.
<i>v</i>	relative position of Point: first Point has index 0
<i>x</i>	the Point's X coordinate.
<i>y</i>	the Point's Y coordinate.
<i>z</i>	the Point's Z coordinate.
<i>m</i>	the Point's M measure.

Returns

0 on failure: any other different value on success.

See also

[gaiaRingGetPoint](#), [gaiaGetPoint](#), [gaiaGetPointXYZ](#), [gaiaSetPointXYM](#), [gaiaSetPointXYZM](#)

Note

this function perform the same identical task performed by [gaiaSetPoint\(\)](#), [gaiaSetPointXYZ\(\)](#), [gaiaSetPointXYM\(\)](#) and [gaiaSetPointXYZM\(\)](#) macros.
 using the [gaiaRingSetPoint\(\)](#) function is a little bit slower but is intrinsically safest, because misused macros can easily cause severe memory corruption.
[gaiaRingSetPoint\(\)](#) instead will always ensure that the appropriate dimensions (as declared by the Ring object) will be correctly used.

6.7.2.118 `gaiaRotateCoords()`

```
GAIAGEO_DECLARE void gaiaRotateCoords (
    gaiaGeomCollPtr geom,
    double angle )
```

Rotates any coordinate within a Geometry object.

Parameters

<i>geom</i>	pointer to Geometry object.
<i>angle</i>	rotation angle [expressed in Degrees].

See also

[gaiaShiftCoords](#), [gaiaScaleCoords](#), [gaiaReflectCoords](#), [gaiaSwapCoords](#)

6.7.2.119 `gaiaSanitize()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSanitize (
    gaiaGeomCollPtr org )
```

Attempts to sanitize a possibly malformed Geometry object.

Parameters

<i>org</i>	pointer to Geometry object.
------------	-----------------------------

Returns

the pointer to newly created Geometry: NULL on failure.

See also

[gaiaIsToxic](#), [gaiaEnsureClosedRings](#), [gaiaRemoveRepeatedPoints](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry created by [gaiaSanitize\(\)](#)

the output Geometry will surely have:

- no repeated Points on Linestrings or Rings (i.e. consecutive Points sharing exactly the same coordinates): any repeated Point will be suppressed, simply leaving only the first occurrence.
- proper Ring closure: for sure any Ring will have exactly coinciding first and last Points.

6.7.2.120 gaiaScaleCoords()

```
GAIA GEO_DECLARE void gaiaScaleCoords (
    gaiaGeomCollPtr geom,
    double scale_x,
    double scale_y )
```

Scales any coordinate within a Geometry object.

Parameters

<i>geom</i>	pointer to Geometry object.
<i>scale_x</i>	X axis scale factor.
<i>scale_y</i>	Y axis scale factor.

See also

[gaiaShiftCoords](#), [gaiaRotateCoords](#), [gaiaReflectCoords](#), [gaiaSwapCoords](#)

6.7.2.121 gaiaShiftCoords()

```
GAIA GEO_DECLARE void gaiaShiftCoords (
    gaiaGeomCollPtr geom,
    double shift_x,
    double shift_y )
```

Shifts any coordinate within a Geometry object.

Parameters

<i>geom</i>	pointer to Geometry object.
<i>shift_x</i>	X axis shift factor.
<i>shift_y</i>	Y axis shift factor.

See also

[gaiaScaleCoords](#), [gaiaRotateCoords](#), [gaiaReflectCoords](#), [gaiaSwapCoords](#), [gaiaShiftCoords3D](#), [gaiaShiftLongitude](#)

6.7.2.122 gaiaShiftCoords3D()

```
GAIA GEO_DECLARE void gaiaShiftCoords3D (
    gaiaGeomCollPtr geom,
```

```
double shift_x,
double shift_y,
double shift_z )
```

Shifts any coordinate within a 3D Geometry object.

Parameters

<i>geom</i>	pointer to Geometry object.
<i>shift</i> ↔ <i>_x</i>	X axis shift factor.
<i>shift</i> ↔ <i>_y</i>	Y axis shift factor.
<i>shift</i> ↔ <i>_z</i>	Z axis shift factor.

See also

[gaiaScaleCoords](#), [gaiaRotateCoords](#), [gaiaReflectCoords](#), [gaiaSwapCoords](#), [gaiaShiftCoords](#), [gaiaShiftLongitude](#), [gaiaNormalizeLonLat](#)

6.7.2.123 **gaiaShiftLongitude()**

```
GAIAGEO_DECLARE void gaiaShiftLongitude (
    gaiaGeomCollPtr geom )
```

Shifts negative longitudes.

Parameters

<i>geom</i>	pointer to Geometry object.
-------------	-----------------------------

See also

[gaiaShiftCoords](#), [gaiaShiftCoords3D](#), [gaiaNormalizeLonLat](#)

Note

only intended for geographic (longitude/latitude) coordinates. Negative longitudes (-180/0) will be shifted by 360, thus allowing to represent longitudes in the 0/360 range and effectively crossing the International Date Line.

6.7.2.124 **gaiaSwapCoords()**

```
GAIAGEO_DECLARE void gaiaSwapCoords (
    gaiaGeomCollPtr geom )
```

Swaps any coordinate within a Geometry object.

Parameters

<i>geom</i>	pointer to Geometry object.
-------------	-----------------------------

See also

[gaiaShiftCoords](#), [gaiaScaleCoords](#), [gaiaRotateCoords](#), [gaiaReflectCoords](#)

Note

the X and Y axes will be swapped.

6.7.2.125 gaiaTrajectoryInterpolatePoint()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaTrajectoryInterpolatePoint (
    gaiaGeomCollPtr geom,
    double m )
```

Attempts to interpolate a Point along a Trajectory accordingly to given M-Value.

Parameters

<i>geom</i>	pointer to Geometry object (expected to be a valid Trajectory)
<i>m</i>	the M-Value to be interpolated

Returns

the pointer to newly created Geometry object representing a Point laying on the input Geometry and positioned at the given M-Value NULL on failure.

See also

[gaiaIsValidTrajectory](#), [gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by [gaiaTrajectoryInterpolatePoint\(\)](#)
not reentrant and thread unsafe.

a Geometry is considered to be a valid Trajectory if it contains a simple LINESTRING supporting M-values growing from each vertex to the next.

6.7.2.126 `gaiaUpDownHeight()`

```
GAIA GEO_DECLARE void gaiaUpDownHeight (
    gaiaLinestringPtr line,
    double * up,
    double * down )
```

Computes the Uphill and Downhill total Height for some 3D Linestrings.

Parameters

<i>line</i>	a generic Linestring.
<i>up</i>	on completion this variable will contain the total Uphill Height. Will always be ZERO for any 2D Linestring.
<i>down</i>	on completion this variable will contain the total Downhill Height. Will always be ZERO for any 2D Linestring.

6.8 src/headers/spatialite/gg_dxf.h File Reference

Geometry handling functions: DXF files.

Data Structures

- struct [gaia_dxf_extra_attr](#)
wrapper for DXF Extra Attribute object
- struct [gaia_dxf_insert](#)
wrapper for DXF Insert object
- struct [gaia_dxf_text](#)
wrapper for DXF Text object
- struct [gaia_dxf_point](#)
wrapper for DXF Point object
- struct [gaia_dxf_circle](#)
wrapper for DXF Circle object
- struct [gaia_dxf_arc](#)
wrapper for DXF Arc object
- struct [gaia_dxf_hole](#)
wrapper for DXF Polygon interior hole object
- struct [gaia_dxf_polyline](#)
wrapper for DXF Polyline object could be a Linestring or a Polygon depending on the is_closed flag
- struct [gaia_dxf_hatch_segm](#)
wrapper for DXF Pattern Segment object
- struct [gaia_dxf_boundary_path](#)
wrapper for DXF Boundary Path object
- struct [gaia_dxf_hatch](#)
wrapper for DXF Pattern Hatch object
- struct [gaia_dxf_block](#)
wrapper for DXF Block object
- struct [gaia_dxf_layer](#)
wrapper for DXF Layer object
- struct [gaia_dxf_parser](#)
wrapper for DXF Parser object
- struct [gaia_dxf_write](#)
wrapper for DXF Write object

Macros

- #define [GAIA_DXF_IMPORT_BY_LAYER](#) 1
import distinct layers
- #define [GAIA_DXF_IMPORT_MIXED](#) 2
import layers mixed altogether by type
- #define [GAIA_DXF_AUTO_2D_3D](#) 3
auto-selects 2D or 3D
- #define [GAIA_DXF_FORCE_2D](#) 4
always force 2D
- #define [GAIA_DXF_FORCE_3D](#) 5
always force 3D
- #define [GAIA_DXF_RING_NONE](#) 6
don't apply any special Ring handling
- #define [GAIA_DXF_RING_LINKED](#) 7
apply special "linked rings" handling
- #define [GAIA_DXF_RING_UNLINKED](#) 8
apply special "unlinked rings" handling
- #define [GAIA_DXF_V12](#) 1000
DXF version [Writer].

Typedefs

- typedef struct [gaia_dxf_extra_attr](#) [gaiaDxfExtraAttr](#)
wrapper for DXF Extra Attribute object
- typedef [gaiaDxfExtraAttr](#) * [gaiaDxfExtraAttrPtr](#)
Typedef for DXF Extra Attribute object.
- typedef struct [gaia_dxf_insert](#) [gaiaDxfInsert](#)
wrapper for DXF Insert object
- typedef [gaiaDxfInsert](#) * [gaiaDxfInsertPtr](#)
Typedef for DXF Insert object.
- typedef struct [gaia_dxf_text](#) [gaiaDxfText](#)
wrapper for DXF Text object
- typedef [gaiaDxfText](#) * [gaiaDxfTextPtr](#)
Typedef for DXF Text object.
- typedef struct [gaia_dxf_point](#) [gaiaDxfPoint](#)
wrapper for DXF Point object
- typedef [gaiaDxfPoint](#) * [gaiaDxfPointPtr](#)
Typedef for DXF Point object.
- typedef struct [gaia_dxf_circle](#) [gaiaDxfCircle](#)
wrapper for DXF Circle object
- typedef [gaiaDxfCircle](#) * [gaiaDxfCirclePtr](#)
Typedef for DXF Circle object.
- typedef struct [gaia_dxf_arc](#) [gaiaDxfArc](#)
wrapper for DXF Arc object
- typedef [gaiaDxfArc](#) * [gaiaDxfArcPtr](#)
Typedef for DXF Arc object.
- typedef struct [gaia_dxf_hole](#) [gaiaDxfHole](#)
wrapper for DXF Polygon interior hole object
- typedef [gaiaDxfHole](#) * [gaiaDxfHolePtr](#)

- Typedef for DXF Point object.*
- typedef struct [gaia_dxf_polyline](#) [gaiaDxfPolyline](#)
wrapper for DXF Polyline object could be a Linestring or a Polygon depending on the is_closed flag
- typedef [gaiaDxfPolyline](#) * [gaiaDxfPolylinePtr](#)
Typedef for DXF Polyline object.
- typedef struct [gaia_dxf_hatch_segm](#) [gaiaDxfHatchSegm](#)
wrapper for DXF Pattern Segment object
- typedef [gaiaDxfHatchSegm](#) * [gaiaDxfHatchSegmPtr](#)
Typedef for DXF Hatch Segment object.
- typedef struct [gaia_dxf_boundary_path](#) [gaiaDxfBoundaryPath](#)
wrapper for DXF Boundary Path object
- typedef [gaiaDxfBoundaryPath](#) * [gaiaDxfBoundaryPathPtr](#)
Typedef for DXF Boundary Path object.
- typedef struct [gaia_dxf_hatch](#) [gaiaDxfHatch](#)
wrapper for DXF Pattern Hatch object
- typedef [gaiaDxfHatch](#) * [gaiaDxfHatchPtr](#)
Typedef for DXF Hatch object.
- typedef struct [gaia_dxf_block](#) [gaiaDxfBlock](#)
wrapper for DXF Block object
- typedef [gaiaDxfBlock](#) * [gaiaDxfBlockPtr](#)
Typedef for DXF Block object.
- typedef struct [gaia_dxf_layer](#) [gaiaDxfLayer](#)
wrapper for DXF Layer object
- typedef [gaiaDxfLayer](#) * [gaiaDxfLayerPtr](#)
Typedef for DXF Layer object.
- typedef struct [gaia_dxf_parser](#) [gaiaDxfParser](#)
wrapper for DXF Parser object
- typedef [gaiaDxfParser](#) * [gaiaDxfParserPtr](#)
Typedef for DXF Layer object.
- typedef struct [gaia_dxf_write](#) [gaiaDxfWriter](#)
wrapper for DXF Write object
- typedef [gaiaDxfWriter](#) * [gaiaDxfWriterPtr](#)
Typedef for DXF Writer object.

Functions

- GAIAGEO_DECLARE [gaiaDxfParserPtr](#) [gaiaCreateDxfParser](#) (int srid, int force_dims, const char *prefix, const char *selected_layer, int special_rings)
Creates a DXF Parser object.
- GAIAGEO_DECLARE void [gaiaDestroyDxfParser](#) ([gaiaDxfParserPtr](#) parser)
Destroying a DXF Parser object.
- GAIAGEO_DECLARE int [gaiaParseDxfFile](#) ([gaiaDxfParserPtr](#) parser, const char *dxf_path)
Parsing a DXF file.
- GAIAGEO_DECLARE int [gaiaParseDxfFile_r](#) (const void *p_cache, [gaiaDxfParserPtr](#) parser, const char *dxf_path)
Parsing a DXF file.
- GAIAGEO_DECLARE int [gaiaLoadFromDxfParser](#) (sqlite3 *db_handle, [gaiaDxfParserPtr](#) parser, int mode, int append)
Populating a DB so to permanently store all Geometries from a DXF Parser.
- GAIAGEO_DECLARE int [gaiaDxfWriterInit](#) ([gaiaDxfWriterPtr](#) dxf, FILE *out, int precision, int version)

- Initializing a DXF Writer Object.*
- GAIAGEO_DECLARE int [gaiaDxfWriteHeader](#) ([gaiaDxfWriterPtr](#) dxf, double minx, double miny, double minz, double maxx, double maxy, double maxz)
- Writing the DXF Header.*
- GAIAGEO_DECLARE int [gaiaDxfWriteFooter](#) ([gaiaDxfWriterPtr](#) dxf)
- Writing a DXF Entities Section Header.*
- GAIAGEO_DECLARE int [gaiaDxfWriteTables](#) ([gaiaDxfWriterPtr](#) dxf)
- Writing the DXF Tables Section Header.*
- GAIAGEO_DECLARE int [gaiaDxfWriteLayer](#) ([gaiaDxfWriterPtr](#) dxf, const char *layer_name)
- Writing a DXF Table/Layer definition.*
- GAIAGEO_DECLARE int [gaiaDxfWriteEntities](#) ([gaiaDxfWriterPtr](#) dxf)
- Writing a DXF Entities Section Header.*
- GAIAGEO_DECLARE int [gaiaDxfWriteEndSection](#) ([gaiaDxfWriterPtr](#) dxf)
- Writing a DXF Entities Section Header.*
- GAIAGEO_DECLARE int [gaiaDxfWritePoint](#) ([gaiaDxfWriterPtr](#) dxf, const char *layer_name, double x, double y, double z)
- Writing a DXF Point Entity.*
- GAIAGEO_DECLARE int [gaiaDxfWriteText](#) ([gaiaDxfWriterPtr](#) dxf, const char *layer_name, double x, double y, double z, const char *label, double text_height, double angle)
- Writing a DXF Text Entity.*
- GAIAGEO_DECLARE int [gaiaDxfWriteLine](#) ([gaiaDxfWriterPtr](#) dxf, const char *layer_name, [gaiaLinestringPtr](#) line)
- Writing a DXF Polyline (opened) Entity.*
- GAIAGEO_DECLARE int [gaiaDxfWriteRing](#) ([gaiaDxfWriterPtr](#) dxf, const char *layer_name, [gaiaRingPtr](#) ring)
- Writing a DXF Polyline (closed) Entity.*
- GAIAGEO_DECLARE int [gaiaDxfWriteGeometry](#) ([gaiaDxfWriterPtr](#) dxf, const char *layer_name, const char *label, double text_height, double text_rotation, [gaiaGeomCollPtr](#) geometry)
- Writing a DXF generic Entity.*
- GAIAGEO_DECLARE int [gaiaExportDxf](#) ([gaiaDxfWriterPtr](#) dxf, sqlite3 *db_handle, const char *sql, const char *layer_col_name, const char *geom_col_name, const char *label_col_name, const char *text_height_col_name, const char *text_rotation_col_name, [gaiaGeomCollPtr](#) geom_filter)
- Exporting a complex DXF file.*

6.8.1 Detailed Description

Geometry handling functions: DXF files.

6.8.2 Typedef Documentation

6.8.2.1 [gaiaDxfArcPtr](#)

```
typedef gaiaDxfArc\* gaiaDxfArcPtr
```

Typedef for DXF Arc object.

See also

[gaiaDxfArc](#)

6.8.2.2 `gaiaDxfBlockPtr`

```
typedef gaiaDxfBlock\* gaiaDxfBlockPtr
```

Typedef for DXF Block object.

See also

[gaiaDxfBlock](#)

6.8.2.3 `gaiaDxfBoundaryPathPtr`

```
typedef gaiaDxfBoundaryPath\* gaiaDxfBoundaryPathPtr
```

Typedef for DXF Boundary Path object.

See also

[gaiaDxfBoundaryPath](#)

6.8.2.4 `gaiaDxfCirclePtr`

```
typedef gaiaDxfCircle\* gaiaDxfCirclePtr
```

Typedef for DXF Circle object.

See also

[gaiaDxfCircle](#)

6.8.2.5 `gaiaDxfExtraAttrPtr`

```
typedef gaiaDxfExtraAttr\* gaiaDxfExtraAttrPtr
```

Typedef for DXF Extra Attribute object.

See also

[gaiaDxfExtraAttr](#)

6.8.2.6 `gaiaDxfHatchPtr`

```
typedef gaiaDxfHatch\* gaiaDxfHatchPtr
```

Typedef for DXF Hatch object.

See also

[gaiaDxfHatch](#)

6.8.2.7 `gaiaDxfHatchSegmPtr`

```
typedef gaiaDxfHatchSegm\* gaiaDxfHatchSegmPtr
```

Typedef for DXF Hatch Segment object.

See also

[gaiaDxfHatch](#)

6.8.2.8 `gaiaDxfHolePtr`

```
typedef gaiaDxfHole\* gaiaDxfHolePtr
```

Typedef for DXF Point object.

See also

[gaiaDxfHole](#)

6.8.2.9 `gaiaDxfInsertPtr`

```
typedef gaiaDxfInsert\* gaiaDxfInsertPtr
```

Typedef for DXF Insert object.

See also

[gaiaDxfText](#)

6.8.2.10 `gaiaDxfLayerPtr`

```
typedef gaiaDxfLayer\* gaiaDxfLayerPtr
```

Typedef for DXF Layer object.

See also

[gaiaDxfLayer](#)

6.8.2.11 `gaiaDxfParserPtr`

```
typedef gaiaDxfParser\* gaiaDxfParserPtr
```

Typedef for DXF Layer object.

See also

[gaiaDxfParser](#)

6.8.2.12 `gaiaDxfPointPtr`

```
typedef gaiaDxfPoint\* gaiaDxfPointPtr
```

Typedef for DXF Point object.

See also

[gaiaDxfPoint](#)

6.8.2.13 `gaiaDxfPolylinePtr`

```
typedef gaiaDxfPolyline\* gaiaDxfPolylinePtr
```

Typedef for DXF Polyline object.

See also

[gaiaDxfPolyline](#)

6.8.2.14 `gaiaDxfTextPtr`

```
typedef gaiaDxfText\* gaiaDxfTextPtr
```

Typedef for DXF Text object.

See also

[gaiaDxfText](#)

6.8.3 Function Documentation

6.8.3.1 `gaiaCreateDxfParser()`

```
GAIAGEO_DECLARE gaiaDxfParserPtr gaiaCreateDxfParser (
    int srid,
    int force_dims,
    const char * prefix,
    const char * selected_layer,
    int special_rings )
```

Creates a DXF Parser object.

Parameters

<i>srid</i>	the SRID value to be used for all Geometries
<i>force_dims</i>	should be one of GAIA_DXF_AUTO_2D_3D, GAIA_DXF_FORCE_2D or GAIA_DXF_FORCE_3D
<i>prefix</i>	an optional prefix to be used for DB target tables (could be NULL)
<i>selected_layers</i>	if set, only the DXF Layer of corresponding name will be imported (could be NULL)
<i>special_rings</i>	rings handling: should be one of GAIA_DXF_RING_NONE, GAIA_DXF_RING_LINKED of GAIA_DXF_RING_UNLINKED

Returns

the pointer to a DXF Parser object

See also

[gaiaDestroyDxfParser](#), [gaiaParseDxfFile](#), [gaiaLoadFromDxfParser](#)

Note

the DXF Parser object corresponds to dynamically allocated memory: so you are responsible to destroy this object before or later by invoking [gaiaDestroyDxfParser\(\)](#).

6.8.3.2 `gaiaDestroyDxfParser()`

```
GAIA GEO_DECLARE void gaiaDestroyDxfParser (
    gaiaDxfParserPtr parser )
```

Destroying a DXF Parser object.

Parameters

<i>parser</i>	pointer to DXF Parser object
---------------	------------------------------

See also

[gaiaCreateDxfParser](#)

Note

the pointer to the DXF Parser object to be finalized is expected to be the one returned by a previous call to `gaiaCreateDxfParser`.

6.8.3.3 `gaiaDxfWriteEndSection()`

```
GAIA GEO_DECLARE int gaiaDxfWriteEndSection (
    gaiaDxfWriterPtr dxf )
```

Writing a DXF Entities Section Header.

Parameters

<i>dxf</i>	pointer to a properly initialized <code>gaiaDxfWriter</code> object
------------	---

Returns

0 on failure, any other value on success

See also

[gaiaDxfWriteTables](#), [gaiaDxfWriteEntities](#)

6.8.3.4 `gaiaDxfWriteEntities()`

```
GAIA GEO_DECLARE int gaiaDxfWriteEntities (
    gaiaDxfWriterPtr dxf )
```

Writing a DXF Entities Section Header.

Parameters

<i>dx</i>	pointer to a properly initialized <code>gaiaDxfWriter</code> object
-----------	---

Returns

0 on failure, any other value on success

See also

[gaiaDxfWriteHeader](#), [gaiaDxfWriteEndSection](#), [gaiaDxfWritePoint](#), [gaiaDxfWriteText](#), [gaiaDxfWriteLine](#),
[gaiaDxfWriteRing](#), [gaiaDxfWriteGeometry](#)

6.8.3.5 gaiaDxfWriteFooter()

```
GAIAGEO_DECLARE int gaiaDxfWriteFooter (  
    gaiaDxfWriterPtr dx )
```

Writing a DXF Entities Section Header.

Parameters

<i>dx</i>	pointer to a properly initialized <code>gaiaDxfWriter</code> object
-----------	---

Returns

0 on failure, any other value on success

See also

[gaiaDxfWriteHeader](#)

6.8.3.6 gaiaDxfWriteGeometry()

```
GAIAGEO_DECLARE int gaiaDxfWriteGeometry (  
    gaiaDxfWriterPtr dx,  
    const char * layer_name,  
    const char * label,  
    double text_height,  
    double text_rotation,  
    gaiaGeomCollPtr geometry )
```

Writing a DXF generic Entity.

Parameters

<i>dxf</i>	pointer to a properly initialized <code>gaiaDxfWriter</code> object
<i>layer_name</i>	name of the corresponding layer
<i>line</i>	pointer to the internal Ring to be exported into the DXF
<i>label</i>	text string containing the label value (could be NULL)
<i>text_height</i>	only for Text Labels: ignored in any other case.
<i>text_rotation</i>	only for Text Labels: ignored in any other case.

Returns

0 on failure, any other value on success

See also

[gaiaDxfWriteEntities](#), [gaiaDxfWriteEndSection](#), [gaiaDxfWritePoint](#), [gaiaDxfWriteText](#), [gaiaDxfWriteLine](#), [gaiaDxfWriteRing](#)

6.8.3.7 `gaiaDxfWriteHeader()`

```
GAIA_GEO_DECLARE int gaiaDxfWriteHeader (
    gaiaDxfWriterPtr dxf,
    double minx,
    double miny,
    double minz,
    double maxx,
    double maxy,
    double maxz )
```

Writing the DXF Header.

Parameters

<i>dxf</i>	pointer to a properly initialized <code>gaiaDxfWriter</code> object
<i>minx</i>	the minimum X coordinate contained within the DXF
<i>miny</i>	the minimum Y coordinate contained within the DXF
<i>minz</i>	the minimum Z coordinate contained within the DXF
<i>maxx</i>	the maximum X coordinate contained within the DXF
<i>maxy</i>	the maximum Y coordinate contained within the DXF
<i>maxz</i>	the maximum Z coordinate contained within the DXF

Returns

0 on failure, any other value on success

See also

[gaiaDxfWriterInit](#), [gaiaDxfWriteFooter](#), [gaiaDxfWriteTables](#), [gaiaDxfWriteEntities](#)

6.8.3.8 gaiaDxfWriteLayer()

```
GAIO_GEO_DECLARE int gaiaDxfWriteLayer (
    gaiaDxfWriterPtr dxf,
    const char * layer_name )
```

Writing a DXF Table/Layer definition.

Parameters

<i>dxf</i>	pointer to a properly initialized gaiaDxfWriter object
<i>layer_name</i>	name of the layer

Returns

0 on failure, any other value on success

See also

[gaiaDxfWriteTables](#), [gaiaDxfWriteEndSection](#)

6.8.3.9 gaiaDxfWriteLine()

```
GAIO_GEO_DECLARE int gaiaDxfWriteLine (
    gaiaDxfWriterPtr dxf,
    const char * layer_name,
    gaiaLinestringPtr line )
```

Writing a DXF Polyline (opened) Entity.

Parameters

<i>dxf</i>	pointer to a properly initialized gaiaDxfWriter object
<i>layer_name</i>	name of the corresponding layer
<i>line</i>	pointer to the internal Linestring to be exported into the DXF

Returns

0 on failure, any other value on success

See also

[gaiaDxfWriteEntities](#), [gaiaDxfWriteEndSection](#), [gaiaDxfWritePoint](#), [gaiaDxfWriteText](#), [gaiaDxfWriteRing](#), [gaiaDxfWriteGeometry](#)

6.8.3.10 gaiaDxfWritePoint()

```
GAIA GEO_DECLARE int gaiaDxfWritePoint (
    gaiaDxfWriterPtr dxf,
    const char * layer_name,
    double x,
    double y,
    double z )
```

Writing a DXF Point Entity.

Parameters

<i>dxf</i>	pointer to a properly initialized gaiaDxfWriter object
<i>layer_name</i>	name of the corresponding layer
<i>x</i>	X coordinate value
<i>y</i>	Y coordinate value
<i>z</i>	Z coordinate value

Returns

0 on failure, any other value on success

See also

[gaiaDxfWriteEntities](#), [gaiaDxfWriteEndSection](#), [gaiaDxfWriteText](#), [gaiaDxfWriteLine](#), [gaiaDxfWriteRing](#), [gaiaDxfWriteGeometry](#)

6.8.3.11 gaiaDxfWriteRing()

```
GAIA GEO_DECLARE int gaiaDxfWriteRing (
    gaiaDxfWriterPtr dxf,
    const char * layer_name,
    gaiaRingPtr ring )
```

Writing a DXF Polyline (closed) Entity.

Parameters

<i>dxf</i>	pointer to a properly initialized gaiaDxfWriter object
<i>layer_name</i>	name of the corresponding layer
<i>line</i>	pointer to the internal Ring to be exported into the DXF

Returns

0 on failure, any other value on success

See also

[gaiaDxfWriteEntities](#), [gaiaDxfWriteEndSection](#), [gaiaDxfWritePoint](#), [gaiaDxfWriteText](#), [gaiaDxfWriteLine](#), [gaiaDxfWriteGeometry](#)

6.8.3.12 gaiaDxfWriterInit()

```
GAIOGEO_DECLARE int gaiaDxfWriterInit (
    gaiaDxfWriterPtr dxf,
    FILE * out,
    int precision,
    int version )
```

Initializing a DXF Writer Object.

Parameters

<i>writer</i>	pointer to the gaiaDxfWriter object to be initialized
<i>out</i>	file handle to DXF output file
<i>precision</i>	number of decimal digits for any coordinate
<i>version</i>	currently always expected to be GAIA_DXF_V12

Returns

0 on failure, any other value on success

See also

[gaiaDxfWriteHeader](#), [gaiaExportDxf](#)

6.8.3.13 gaiaDxfWriteTables()

```
GAIOGEO_DECLARE int gaiaDxfWriteTables (
    gaiaDxfWriterPtr dxf )
```

Writing the DXF Tables Section Header.

Parameters

<i>dxf</i>	pointer to a properly initialized gaiaDxfWriter object
------------	--

Returns

0 on failure, any other value on success

See also

[gaiaDxfWriteHeader](#), [gaiaDxfWriteEndSection](#)

6.8.3.14 `gaiaDxfWriteText()`

```
GAIA GEO_DECLARE int gaiaDxfWriteText (
    gaiaDxfWriterPtr dxf,
    const char * layer_name,
    double x,
    double y,
    double z,
    const char * label,
    double text_height,
    double angle )
```

Writing a DXF Text Entity.

Parameters

<i>dxf</i>	pointer to a properly initialized <code>gaiaDxfWriter</code> object
<i>layer_name</i>	name of the corresponding layer
<i>x</i>	X coordinate value
<i>y</i>	Y coordinate value
<i>z</i>	Z coordinate value
<i>label</i>	text string containing the label value
<i>text_height</i>	height of the text in map units
<i>angle</i>	text rotation angle

Returns

0 on failure, any other value on success

See also

[gaiaDxfWriteEntities](#), [gaiaDxfWriteEndSection](#), [gaiaDxfWritePoint](#), [gaiaDxfWriteLine](#), [gaiaDxfWriteRing](#), [gaiaDxfWriteGeometry](#)

6.8.3.15 `gaiaExportDxf()`

```
GAIA GEO_DECLARE int gaiaExportDxf (
    gaiaDxfWriterPtr dxf,
    sqlite3 * db_handle,
    const char * sql,
    const char * layer_col_name,
    const char * geom_col_name,
```

```
const char * label_col_name,  
const char * text_height_col_name,  
const char * text_rotation_col_name,  
gaiaGeomCollPtr geom_filter )
```

Exporting a complex DXF file.

Parameters

<i>dxf</i>	pointer to a properly initialized gaiaDxfWriter object
<i>db_handle</i>	handle to the current DB connection
<i>sql</i>	a text string defining the SQL query to be used for extracting all geometries/entities to be exported into the output DXF
<i>layer_col_name</i>	name of the SQL resultset column containing the Layer name
<i>geom_col_name</i>	name of the SQL resultset column containing Geometries
<i>label_col_name</i>	name of the SQL resultset column containing Label values (could be NULL)
<i>text_height_col_name</i>	name of the SQL resultset column containing Text Height values (could be NULL)
<i>text_rotation_col_name</i>	name of the SQL resultset column containing Text Rotation values (could be NULL)
<i>geom_filter</i>	an optional arbitrary Geometry to be used as a Spatial Filter (could be NULL)

Returns

0 on failure; the total count of exported entities on success

See also

[gaiaDxfWriterInit](#)

6.8.3.16 gaiaLoadFromDxfParser()

```
GAIA_GEO_DECLARE int gaiaLoadFromDxfParser (
    sqlite3 * db_handle,
    gaiaDxfParserPtr parser,
    int mode,
    int append )
```

Populating a DB so to permanently store all Geometries from a DXF Parser.

Parameters

<i>db_handle</i>	handle to a valid DB connection
<i>parser</i>	pointer to DXF Parser object
<i>mode</i>	should be one of GAIA_DXF_IMPORT_BY_LAYER or GAIA_DXF_IMPORT_MIXED
<i>append</i>	boolean flag: if set and some required DB table already exists will attempt to append further rows into the existing table. otherwise an error will be returned.

Returns

0 on failure, any other value on success

See also

[gaiaCreateDxfParser](#), [gaiaDestroyDxfParser](#), [gaiaParseDxfFile](#)

Note

the pointer to the DXF Parser object is expected to be the one returned by a previous call to `gaiaCreateDxfParser` and previously used for a succesfull call to `gaiaParseDxfFile`

6.8.3.17 `gaiaParseDxfFile()`

```
GAIA GEO_DECLARE int gaiaParseDxfFile (
    gaiaDxfParserPtr parser,
    const char * dxf_path )
```

Parsing a DXF file.

Parameters

<i>parser</i>	pointer to DXF Parser object
<i>dxf_path</i>	pathname of the DXF external file to be parsed

Returns

0 on failure, any other value on success

See also

[gaiaParseDxfFile_r](#), [gaiaCreateDxfParser](#), [gaiaDestroyDxfParser](#), [gaiaLoadFromDxfParser](#)

Note

the pointer to the DXF Parser object is expected to be the one returned by a previous call to `gaiaCreateDxfParser`. A DXF Parser object can be used only a single time to parse a DXF file.
not reentrant and thread unsafe.

6.8.3.18 `gaiaParseDxfFile_r()`

```
GAIA GEO_DECLARE int gaiaParseDxfFile_r (
    const void * p_cache,
    gaiaDxfParserPtr parser,
    const char * dxf_path )
```

Parsing a DXF file.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>parser</i>	pointer to DXF Parser object
<i>dxf_path</i>	pathname of the DXF external file to be parsed

Returns

0 on failure, any other value on success

See also

[gaiaParseDxfFile](#), [gaiaCreateDxfParser](#), [gaiaDestroyDxfParser](#), [gaiaLoadFromDxfParser](#)

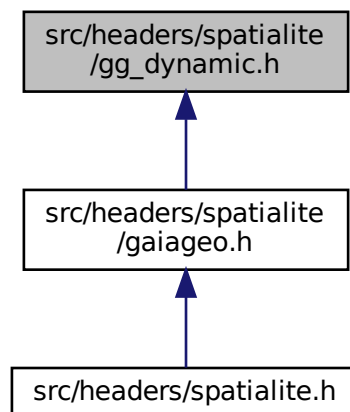
Note

the pointer to the DXF Parser object is expected to be the one returned by a previous call to [gaiaCreateDxfParser](#). A DXF Parser object can be used only a single time to parse a DXF file. reentrant and thread-safe.

6.9 src/headers/spatialite/gg_dynamic.h File Reference

Geometry handling functions: DynamicLine handling.

This graph shows which files directly or indirectly include this file:

**Functions**

- GAIAGEO_DECLARE [gaiaDynamicLinePtr](#) [gaiaAllocDynamicLine](#) (void)
Creates a new dynamically growing line/ring object.
- GAIAGEO_DECLARE void [gaiaFreeDynamicLine](#) ([gaiaDynamicLinePtr](#) p)
Destroys a dynamically growing line/ring object.
- GAIAGEO_DECLARE [gaiaPointPtr](#) [gaiaAppendPointToDynamicLine](#) ([gaiaDynamicLinePtr](#) p, double x, double y)
Appends a new 2D Point [XY] at the end of a dynamically growing line/ring object.
- GAIAGEO_DECLARE [gaiaPointPtr](#) [gaiaAppendPointZToDynamicLine](#) ([gaiaDynamicLinePtr](#) p, double x, double y, double z)

Appends a new 3D Point [XYZ] at the end of a dynamically growing line/ring object.

- GAIAGEO_DECLARE [gaiaPointPtr](#) [gaiaAppendPointMToDynamicLine](#) ([gaiaDynamicLinePtr](#) p, double x, double y, double m)

Appends a new 2D Point [XYM] at the end of a dynamically growing line/ring object.

- GAIAGEO_DECLARE [gaiaPointPtr](#) [gaiaAppendPointZMToDynamicLine](#) ([gaiaDynamicLinePtr](#) p, double x, double y, double z, double m)

Appends a new 3D Point [XYZM] at the end of a dynamically growing line/ring object.

- GAIAGEO_DECLARE [gaiaPointPtr](#) [gaiaPrependPointToDynamicLine](#) ([gaiaDynamicLinePtr](#) p, double x, double y)

Appends a new 2D Point [XY] before the first one of a dynamically growing line/ring object.

- GAIAGEO_DECLARE [gaiaPointPtr](#) [gaiaPrependPointZToDynamicLine](#) ([gaiaDynamicLinePtr](#) p, double x, double y, double z)

Appends a new 3D Point [XYZ] before the first one of a dynamically growing line/ring object.

- GAIAGEO_DECLARE [gaiaPointPtr](#) [gaiaPrependPointMToDynamicLine](#) ([gaiaDynamicLinePtr](#) p, double x, double y, double m)

Appends a new 2D Point [XYM] before the first one of a dynamically growing line/ring object.

- GAIAGEO_DECLARE [gaiaPointPtr](#) [gaiaPrependPointZMToDynamicLine](#) ([gaiaDynamicLinePtr](#) p, double x, double y, double z, double m)

Appends a new 3D Point [XYZM] before the first one of a dynamically growing line/ring object.

- GAIAGEO_DECLARE [gaiaPointPtr](#) [gaiaDynamicLineInsertAfter](#) ([gaiaDynamicLinePtr](#) p, [gaiaPointPtr](#) pt, double x, double y)

Appends a new 2D Point [XY] immediately after the given Point into a dynamically growing line/ring object.

- GAIAGEO_DECLARE [gaiaPointPtr](#) [gaiaDynamicLineInsertBefore](#) ([gaiaDynamicLinePtr](#) p, [gaiaPointPtr](#) pt, double x, double y)

Appends a new 2D Point [XY] immediately before the given Point into a dynamically growing line/ring object.

- GAIAGEO_DECLARE void [gaiaDynamicLineDeletePoint](#) ([gaiaDynamicLinePtr](#) p, [gaiaPointPtr](#) pt)

Removes a given Point from a dynamically growing line/ring object.

- GAIAGEO_DECLARE [gaiaDynamicLinePtr](#) [gaiaCloneDynamicLine](#) ([gaiaDynamicLinePtr](#) org)

Duplicates a dynamically growing line/ring object.

- GAIAGEO_DECLARE [gaiaDynamicLinePtr](#) [gaiaReverseDynamicLine](#) ([gaiaDynamicLinePtr](#) org)

Duplicates and reverts a dynamically growing line/ring object.

- GAIAGEO_DECLARE [gaiaDynamicLinePtr](#) [gaiaDynamicLineSplitBefore](#) ([gaiaDynamicLinePtr](#) org, [gaiaPointPtr](#) point)

Cuts a dynamically growing line/ring in two halves, using a given cut point.

- GAIAGEO_DECLARE [gaiaDynamicLinePtr](#) [gaiaDynamicLineSplitAfter](#) ([gaiaDynamicLinePtr](#) org, [gaiaPointPtr](#) point)

Cuts a dynamically growing line/ring in two halves, using a given cut point.

- GAIAGEO_DECLARE [gaiaDynamicLinePtr](#) [gaiaDynamicLineJoinAfter](#) ([gaiaDynamicLinePtr](#) org, [gaiaPointPtr](#) point, [gaiaDynamicLinePtr](#) toJoin)

Merges two dynamically growing line/ring object into a single one.

- GAIAGEO_DECLARE [gaiaDynamicLinePtr](#) [gaiaDynamicLineJoinBefore](#) ([gaiaDynamicLinePtr](#) org, [gaiaPointPtr](#) point, [gaiaDynamicLinePtr](#) toJoin)

Merges two dynamically growing line/ring object into a single one.

- GAIAGEO_DECLARE [gaiaPointPtr](#) [gaiaDynamicLineFindByCoords](#) ([gaiaDynamicLinePtr](#) p, double x, double y)

Finds a Point within a dynamically growing line/ring object [by coords].

- GAIAGEO_DECLARE [gaiaPointPtr](#) [gaiaDynamicLineFindByPos](#) ([gaiaDynamicLinePtr](#) p, int pos)

Finds a Point within a dynamically growing line/ring object [by position].

- GAIAGEO_DECLARE [gaiaDynamicLinePtr](#) [gaiaCreateDynamicLine](#) (double *coords, int points)

Creates a new dynamically growing line/ring object.

6.9.1 Detailed Description

Geometry handling functions: DynamicLine handling.

6.9.2 Function Documentation

6.9.2.1 `gaiaAllocDynamicLine()`

```
GAIA GEO_DECLARE gaiaDynamicLinePtr gaiaAllocDynamicLine (
    void )
```

Creates a new dynamicly growing line/ring object.

Returns

the pointer to newly created object

See also

[gaiaCreateDynamicLine](#), [gaiaFreeDynamicLine](#)

Note

you are responsible to destroy (before or after) any allocated dynamically growing line/ring object.

6.9.2.2 `gaiaAppendPointMToDynamicLine()`

```
GAIA GEO_DECLARE gaiaPointPtr gaiaAppendPointMToDynamicLine (
    gaiaDynamicLinePtr p,
    double x,
    double y,
    double m )
```

Appends a new 2D Point [XYM] at the end of a dynamically growing line/ring object.

Parameters

<i>p</i>	pointer to the dynamically growing line/ring object.
<i>x</i>	X coordinate of the Point
<i>y</i>	Y coordinate of the Point
<i>m</i>	M measure of the Point

Returns

the pointer to newly created Point

6.9.2.3 gaiaAppendPointToDynamicLine()

```
GAIA GEO_DECLARE gaiaPointPtr gaiaAppendPointToDynamicLine (
    gaiaDynamicLinePtr p,
    double x,
    double y )
```

Appends a new 2D Point [XY] at the end of a dynamically growing line/ring object.

Parameters

<i>p</i>	pointer to the dynamically growing line/ring object.
<i>x</i>	X coordinate of the Point
<i>y</i>	Y coordinate of the Point

Returns

the pointer to newly created Point

6.9.2.4 gaiaAppendPointZMToDynamicLine()

```
GAIA GEO_DECLARE gaiaPointPtr gaiaAppendPointZMToDynamicLine (
    gaiaDynamicLinePtr p,
    double x,
    double y,
    double z,
    double m )
```

Appends a new 3D Point [XYZM] at the end of a dynamically growing line/ring object.

Parameters

<i>p</i>	pointer to the dynamically growing line/ring object.
<i>x</i>	X coordinate of the Point
<i>y</i>	Y coordinate of the Point
<i>z</i>	Z coordinate of the Point
<i>m</i>	M measure of the Point

Returns

the pointer to newly created Point

6.9.2.5 gaiaAppendPointZToDynamicLine()

```
GAIA GEO_DECLARE gaiaPointPtr gaiaAppendPointZToDynamicLine (
    gaiaDynamicLinePtr p,
    double x,
    double y,
    double z )
```

Appends a new 3D Point [XYZ] at the end of a dynamically growing line/ring object.

Parameters

<i>p</i>	pointer to the dynamically growing line/ring object.
<i>x</i>	X coordinate of the Point
<i>y</i>	Y coordinate of the Point
<i>z</i>	Z coordinate of the Point

Returns

the pointer to newly created Point

6.9.2.6 gaiaCloneDynamicLine()

```
GAIA GEO_DECLARE gaiaDynamicLinePtr gaiaCloneDynamicLine (
    gaiaDynamicLinePtr org )
```

Duplicates a dynamically growing line/ring object.

Parameters

<i>org</i>	pointer to dynamically growing line/ring object [origin].
------------	---

Returns

the pointer to newly created dynamic growing line/ring object: NULL on failure.

Note

the newly created object is an exact copy of the original one.

6.9.2.7 gaiaCreateDynamicLine()

```
GAIA GEO_DECLARE gaiaDynamicLinePtr gaiaCreateDynamicLine (
    double * coords,
    int points )
```

Creates a new dynamicly growing line/ring object.

Parameters

<i>coords</i>	an array of COORDs, any dimension [XY, XYZ, XYM, XYZM]
<i>points</i>	number of points [aka vertices] into the array

Returns

the pointer to newly created object

See also

[gaiaAllocDynamicLine](#), [gaiaFreeDynamicLine](#), [gaiaLinestringStruct](#), [gaiaRingStruct](#)

Note

you are responsible to destroy (before or after) any allocated dynamically growing line/ring object.
The COORDs array is usually expected to be one found within a [gaiaLinestring](#) or [gaiaRing](#) object.

6.9.2.8 [gaiaDynamicLineDeletePoint\(\)](#)

```
GAIA GEO_DECLARE void gaiaDynamicLineDeletePoint (
    gaiaDynamicLinePtr p,
    gaiaPointPtr pt )
```

Removes a given Point from a dynamically growing line/ring object.

Parameters

<i>p</i>	pointer to dynamically growing line/ring object.
<i>pt</i>	pointer to given Point.

Note

the given Point (referenced by its address) will be removed from the dynamically growing line/ring object.
the given Point will be then implicitly destroyed.

6.9.2.9 [gaiaDynamicLineFindByCoords\(\)](#)

```
GAIA GEO_DECLARE gaiaPointPtr gaiaDynamicLineFindByCoords (
    gaiaDynamicLinePtr p,
    double x,
    double y )
```

Finds a Point within a dynamically growing line/ring object [by coords].

Parameters

<i>p</i>	pointer to dynamically line/ring object.
<i>x</i>	Point X coordinate.
<i>y</i>	Point Y coordinate.

Returns

the pointer to the corresponding Point object: NULL on failure.

See also

[gaiaDynamicLineFindByPos](#)

Note

if the line object contains more Points sharing the same coordinates, a reference to the first one found will be returned.

6.9.2.10 `gaiaDynamicLineFindByPos()`

```
GAIA_GEO_DECLARE gaiaPointPtr gaiaDynamicLineFindByPos (
    gaiaDynamicLinePtr p,
    int pos )
```

Finds a Point within a dynamically growing line/ring object [by position].

Parameters

<i>p</i>	pointer to dynamically line/ring object.
<i>pos</i>	relative position [first Point has index 0].

Returns

the pointer to the corresponding Point object: NULL on failure.

See also

[gaiaDynamicLineFindByCoords](#)

6.9.2.11 `gaiaDynamicLineInsertAfter()`

```
GAIA_GEO_DECLARE gaiaPointPtr gaiaDynamicLineInsertAfter (
    gaiaDynamicLinePtr p,
```

```

    gaiaPointPtr pt,
    double x,
    double y )

```

Appends a new 2D Point [XY] immediately after the given Point into a dynamically growing line/ring object.

Parameters

<i>p</i>	pointer to the dynamically growing line/ring object.
<i>pt</i>	pointer to the given Point.
<i>x</i>	X coordinate of the Point to be appended
<i>y</i>	Y coordinate of the Point to be appended

See also

gaiaDynamicLiceInsertBefore

Returns

the pointer to newly created Point

6.9.2.12 gaiaDynamicLineInsertBefore()

```

GAIA GEO_DECLARE gaiaPointPtr gaiaDynamicLineInsertBefore (
    gaiaDynamicLinePtr p,
    gaiaPointPtr pt,
    double x,
    double y )

```

Appends a new 2D Point [XY] immediately before the given Point into a dynamically growing line/ring object.

Parameters

<i>p</i>	pointer to the dynamically growing line/ring object.
<i>pt</i>	pointer to the given Point.
<i>x</i>	X coordinate of the Point to be appended
<i>y</i>	Y coordinate of the Point to be appended

See also

gaiaDynamicLiceInsertBeforeAfter

Returns

the pointer to newly created Point

6.9.2.13 `gaiaDynamicLineJoinAfter()`

```
GAIA_GEO_DECLARE gaiaDynamicLinePtr gaiaDynamicLineJoinAfter (
    gaiaDynamicLinePtr org,
    gaiaPointPtr point,
    gaiaDynamicLinePtr toJoin )
```

Merges two dynamically growing line/ring object into a single one.

Parameters

<i>org</i>	pointer to the first input object [first line].
<i>point</i>	pointer to the reference Point object.
<i>toJoin</i>	pointer to the second input object [second line].

Returns

the pointer to newly created dynamically growing line/ring object [merged line]: NULL on failure.

See also

[gaiaDynamicLineJoinBefore](#)

Note

the reference Point must exists into the first line: the second line will then be inserted immediately after the reference Point.

The newly created object will represent the resulting merged line:
both input objects remain untouched.

6.9.2.14 `gaiaDynamicLineJoinBefore()`

```
GAIA_GEO_DECLARE gaiaDynamicLinePtr gaiaDynamicLineJoinBefore (
    gaiaDynamicLinePtr org,
    gaiaPointPtr point,
    gaiaDynamicLinePtr toJoin )
```

Merges two dynamically growing line/ring object into a single one.

Parameters

<i>org</i>	pointer to the first input object [first line].
<i>point</i>	pointer to the reference Point object.
<i>toJoin</i>	pointer to the second input object [second line].

Returns

the pointer to newly created dynamically growing line/ring object [merged line]: NULL on failure.

See also

[gaiaDynamicLineJoinAfter](#)

Note

the reference Point must exists into the first line: the second line will then be inserted immediately before the reference Point.

The newly created object will represent the resulting merged line:

both input objects remain untouched.

6.9.2.15 gaiaDynamicLineSplitAfter()

```
GAIAGEO_DECLARE gaiaDynamicLinePtr gaiaDynamicLineSplitAfter (
    gaiaDynamicLinePtr org,
    gaiaPointPtr point )
```

Cuts a dynamically growing line/ring in two halves, using a given cut point.

Parameters

<i>org</i>	pointer to the input object [the line to be split].
<i>point</i>	pointer to given cut point.

Returns

the pointer to newly created dynamic growing line/ring object: NULL on failure.

See also

[gaiaDynamicLineSplitBefore](#)

Note

the newly created object will contain a line going from the original first point to the cut point [included].

on completion the original line will be reduced, going from the cut point [excluded] to the original last point.

6.9.2.16 gaiaDynamicLineSplitBefore()

```
GAIAGEO_DECLARE gaiaDynamicLinePtr gaiaDynamicLineSplitBefore (
    gaiaDynamicLinePtr org,
    gaiaPointPtr point )
```

Cuts a dynamically growing line/ring in two halves, using a given cut point.

Parameters

<i>org</i>	pointer to the input object [the line to be split].
<i>point</i>	pointer to given cut point.

Returns

the pointer to newly created dynamic growing line/ring object: NULL on failure.

See also

[gaiaDynamicLineSplitAfter](#)

Note

the newly created object will contain a line going from the original first point to the cut point [excluded].
on completion the original line will be reduced, going from the cut point [included] to the original last point.

6.9.2.17 `gaiaFreeDynamicLine()`

```
GAIA GEO_DECLARE void gaiaFreeDynamicLine (
    gaiaDynamicLinePtr p )
```

Destroys a dynamically growing line/ring object.

Parameters

<i>p</i>	pointer to object to be destroyed
----------	-----------------------------------

See also

[gaiaAllocDynamicLine](#)

6.9.2.18 `gaiaPrependPointMToDynamicLine()`

```
GAIA GEO_DECLARE gaiaPointPtr gaiaPrependPointMToDynamicLine (
    gaiaDynamicLinePtr p,
    double x,
    double y,
    double m )
```

Appends a new 2D Point [XYM] before the first one of a dynamically growing line/ring object.

Parameters

<i>p</i>	pointer to the dynamically growing line/ring object.
<i>x</i>	X coordinate of the Point
<i>y</i>	Y coordinate of the Point
<i>m</i>	M measure of the Point

Returns

the pointer to newly created Point

6.9.2.19 gaiaPrependPointToDynamicLine()

```
GAIA GEO_DECLARE gaiaPointPtr gaiaPrependPointToDynamicLine (
    gaiaDynamicLinePtr p,
    double x,
    double y )
```

Appends a new 2D Point [XY] before the first one of a dynamically growing line/ring object.

Parameters

<i>p</i>	pointer to the dynamically growing line/ring object.
<i>x</i>	X coordinate of the Point
<i>y</i>	Y coordinate of the Point

Returns

the pointer to newly created Point

6.9.2.20 gaiaPrependPointZMToDynamicLine()

```
GAIA GEO_DECLARE gaiaPointPtr gaiaPrependPointZMToDynamicLine (
    gaiaDynamicLinePtr p,
    double x,
    double y,
    double z,
    double m )
```

Appends a new 3D Point [XYZM] before the first one of a dynamically growing line/ring object.

Parameters

<i>p</i>	pointer to the dynamically growing line/ring object.
<i>x</i>	X coordinate of the Point
<i>y</i>	Y coordinate of the Point
<i>z</i>	Z coordinate of the Point
<i>m</i>	M measure of the Point

Returns

the pointer to newly created Point

6.9.2.21 gaiaPrependPointZToDynamicLine()

```
GAIA_GEO_DECLARE gaiaPointPtr gaiaPrependPointZToDynamicLine (
    gaiaDynamicLinePtr p,
    double x,
    double y,
    double z )
```

Appends a new 3D Point [XYZ] before the first one of a dynamically growing line/ring object.

Parameters

<i>p</i>	pointer to the dynamically growing line/ring object.
<i>x</i>	X coordinate of the Point
<i>y</i>	Y coordinate of the Point
<i>z</i>	Z coordinate of the Point

Returns

the pointer to newly created Point

6.9.2.22 gaiaReverseDynamicLine()

```
GAIA_GEO_DECLARE gaiaDynamicLinePtr gaiaReverseDynamicLine (
    gaiaDynamicLinePtr org )
```

Duplicates and reverts a dynamically growing line/ring object.

Parameters

<i>org</i>	pointer to dynamically growing line/ring object [origin].
------------	---

Returns

the pointer to newly created dynamic growing line/ring object: NULL on failure.

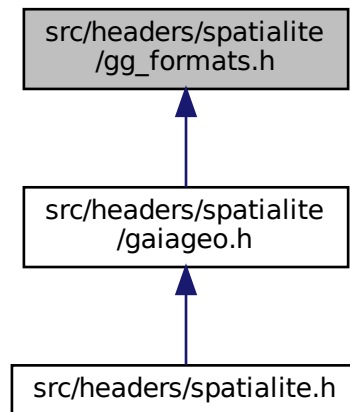
Note

the newly created object is an exact copy of the original one, except in that direction is reverted.
i.e. first input point becomes last output point, and last input point becomes first output point.

6.10 src/headers/spatialite/gg_formats.h File Reference

Geometry handling functions: formats.

This graph shows which files directly or indirectly include this file:



Functions

- GAIAGEO_DECLARE int [gaiaEndianArch](#) (void)
Test CPU endianness.
- GAIAGEO_DECLARE short [gaiaImport16](#) (const unsigned char *p, int little_endian, int little_endian_arch)
Import an INT-16 value in endian-aware fashion.
- GAIAGEO_DECLARE int [gaiaImport32](#) (const unsigned char *p, int little_endian, int little_endian_arch)
Import an INT-32 value in endian-aware fashion.
- GAIAGEO_DECLARE unsigned int [gaiaImportU32](#) (const unsigned char *p, int little_endian, int little_endian_arch)
Import an UINT-32 value in endian-aware fashion.
- GAIAGEO_DECLARE float [gaiaImportF32](#) (const unsigned char *p, int little_endian, int little_endian_arch)
Import a FLOAT-32 value in endian-aware fashion.
- GAIAGEO_DECLARE double [gaiaImport64](#) (const unsigned char *p, int little_endian, int little_endian_arch)
Import an DOUBLE-64 in endian-aware fashion.
- GAIAGEO_DECLARE sqlite3_int64 [gaiaImportI64](#) (const unsigned char *p, int little_endian, int little_endian_arch)
Import an INT-64 in endian-aware fashion.
- GAIAGEO_DECLARE void [gaiaExport16](#) (unsigned char *p, short value, int little_endian, int little_endian_arch)
Export an INT-16 value in endian-aware fashion.
- GAIAGEO_DECLARE void [gaiaExport32](#) (unsigned char *p, int value, int little_endian, int little_endian_arch)
Export an INT-32 value in endian-aware fashion.
- GAIAGEO_DECLARE void [gaiaExportU32](#) (unsigned char *p, unsigned int value, int little_endian, int little_endian_arch)
Export an UINT-32 value in endian-aware fashion.

- GAIAGEO_DECLARE void [gaiaExportF32](#) (unsigned char *p, float value, int little_endian, int little_endian↵_arch)
Export a FLOAT-32 value in endian-aware fashion.
- GAIAGEO_DECLARE void [gaiaExport64](#) (unsigned char *p, double value, int little_endian, int little_endian↵_arch)
Export a DOUBLE value in endian-aware fashion.
- GAIAGEO_DECLARE void [gaiaExportI64](#) (unsigned char *p, sqlite3_int64 value, int little_endian, int little_↵_arch)
Export an INT-64 value in endian-aware fashion.
- GAIAGEO_DECLARE void [gaiaOutBufferInitialize](#) ([gaiaOutBufferPtr](#) buf)
Initializes a dynamically growing Text output buffer.
- GAIAGEO_DECLARE void [gaiaOutBufferReset](#) ([gaiaOutBufferPtr](#) buf)
Resets a dynamically growing Text output buffer to its initial (empty) state.
- GAIAGEO_DECLARE void [gaiaAppendToOutBuffer](#) ([gaiaOutBufferPtr](#) buf, const char *text)
Appends a text string at the end of Text output buffer.
- GAIAGEO_DECLARE void [gaiaMakePoint](#) (double x, double y, int srid, unsigned char **result, int *size)
Creates a BLOB-Geometry representing a Point (BLOB-Geometry)
- GAIAGEO_DECLARE void [gaiaMakePointZ](#) (double x, double y, double z, int srid, unsigned char **result, int *size)
Creates a BLOB-Geometry representing a PointZ (BLOB-Geometry)
- GAIAGEO_DECLARE void [gaiaMakePointM](#) (double x, double y, double m, int srid, unsigned char **result, int *size)
Creates a BLOB-Geometry representing a PointM (BLOB-Geometry)
- GAIAGEO_DECLARE void [gaiaMakePointZM](#) (double x, double y, double z, double m, int srid, unsigned char **result, int *size)
Creates a BLOB-Geometry representing a PointZM (BLOB-Geometry)
- GAIAGEO_DECLARE void [gaiaMakePointEx](#) (int tiny_point, double x, double y, int srid, unsigned char **result, int *size)
Creates a BLOB-Geometry representing a Point (BLOB-Geometry or BLOB-TinyPoint)
- GAIAGEO_DECLARE void [gaiaMakePointZEx](#) (int tiny_point, double x, double y, double z, int srid, unsigned char **result, int *size)
Creates a BLOB-Geometry representing a PointZ (BLOB-Geometry or BLOB-TinyPoint)
- GAIAGEO_DECLARE void [gaiaMakePointMEx](#) (int tiny_point, double x, double y, double m, int srid, unsigned char **result, int *size)
Creates a BLOB-Geometry representing a PointM (BLOB-Geometry or BLOB-TinyPoint)
- GAIAGEO_DECLARE void [gaiaMakePointZMEx](#) (int tiny_point, double x, double y, double z, double m, int srid, unsigned char **result, int *size)
Creates a BLOB-Geometry representing a PointZM (BLOB-Geometry or BLOB-TinyPoint)
- GAIAGEO_DECLARE void [gaiaMakeLine](#) ([gaiaGeomCollPtr](#) geom1, [gaiaGeomCollPtr](#) geom2, unsigned char **result, int *size)
Creates a BLOB-Geometry representing a Segment (2-Points Linestring)
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaFromSpatialLiteBlobWkb](#) (const unsigned char *blob, unsigned int size)
Creates a Geometry object from the corresponding BLOB-Geometry.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaFromSpatialLiteBlobWkbEx](#) (const unsigned char *blob, unsigned int size, int gpkg_mode, int gpkg_amphibious)
Creates a Geometry object from the corresponding BLOB-Geometry.
- GAIAGEO_DECLARE void [gaiaToSpatialLiteBlobWkb](#) ([gaiaGeomCollPtr](#) geom, unsigned char **result, int *size)
Creates a BLOB-Geometry corresponding to a Geometry object.
- GAIAGEO_DECLARE void [gaiaToSpatialLiteBlobWkbEx](#) ([gaiaGeomCollPtr](#) geom, unsigned char **result, int *size, int gpkg_mode)
Creates a BLOB-Geometry corresponding to a Geometry object.

- GAIAGEO_DECLARE void [gaiaToSpatialiteBlobWkbEx2](#) ([gaiaGeomCollPtr](#) geom, unsigned char **result, int *size, int gpkg_mode, int tiny_point)
Creates a BLOB-Geometry corresponding to a Geometry object.
- GAIAGEO_DECLARE void [gaiaToCompressedBlobWkb](#) ([gaiaGeomCollPtr](#) geom, unsigned char **result, int *size)
Creates a Compressed BLOB-Geometry corresponding to a Geometry object.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaFromWkb](#) (const unsigned char *blob, unsigned int size)
Creates a Geometry object from WKB notation.
- GAIAGEO_DECLARE void [gaiaToWkb](#) ([gaiaGeomCollPtr](#) geom, unsigned char **result, int *size)
Encodes a Geometry object into WKB notation.
- GAIAGEO_DECLARE char * [gaiaToHexWkb](#) ([gaiaGeomCollPtr](#) geom)
Encodes a Geometry object into (hex) WKB notation.
- GAIAGEO_DECLARE void [gaiaToEWKB](#) ([gaiaOutBufferPtr](#) out_buf, [gaiaGeomCollPtr](#) geom)
Encodes a Geometry object into EWKB notation.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaFromEWKB](#) (const unsigned char *in_buffer)
Creates a Geometry object from EWKB notation.
- GAIAGEO_DECLARE unsigned char * [gaiaParseHexEWKB](#) (const unsigned char *blob_hex, int *blob_size)
Translates an EWKB notation from hexadecimal into binary.
- GAIAGEO_DECLARE int [gaiaEwkbGetPoint](#) ([gaiaGeomCollPtr](#) geom, unsigned char *blob, int offset, int blob_size, int endian, int endian_arch, int dims)
Attempts to decode a Point from within an EWKB binary buffer.
- GAIAGEO_DECLARE int [gaiaEwkbGetLinestring](#) ([gaiaGeomCollPtr](#) geom, unsigned char *blob, int offset, int blob_size, int endian, int endian_arch, int dims)
Attempts to decode a Point from within an EWKB binary buffer.
- GAIAGEO_DECLARE int [gaiaEwkbGetPolygon](#) ([gaiaGeomCollPtr](#) geom, unsigned char *blob, int offset, int blob_size, int endian, int endian_arch, int dims)
Attempts to decode a Polygon from within an EWKB binary buffer.
- GAIAGEO_DECLARE int [gaiaEwkbGetMultiGeometry](#) ([gaiaGeomCollPtr](#) geom, unsigned char *blob, int offset, int blob_size, int endian, int endian_arch, int dims)
Attempts to decode a MultiGeometry from within an EWKB binary buffer.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaFromFgf](#) (const unsigned char *blob, unsigned int size)
Creates a Geometry object from FGF notation.
- GAIAGEO_DECLARE void [gaiaToFgf](#) ([gaiaGeomCollPtr](#) geom, unsigned char **result, int *size, int coord_dims)
Encodes a Geometry object into FGF notation.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaParseWkt](#) (const unsigned char *in_buffer, short type)
Creates a Geometry object from WKT notation.
- GAIAGEO_DECLARE void [gaiaOutWkt](#) ([gaiaOutBufferPtr](#) out_buf, [gaiaGeomCollPtr](#) geom)
Encodes a Geometry object into WKT notation.
- GAIAGEO_DECLARE void [gaiaOutWktEx](#) ([gaiaOutBufferPtr](#) out_buf, [gaiaGeomCollPtr](#) geom, int precision)
Encodes a Geometry object into WKT notation.
- GAIAGEO_DECLARE void [gaiaOutWktStrict](#) ([gaiaOutBufferPtr](#) out_buf, [gaiaGeomCollPtr](#) geom, int precision)
Encodes a Geometry object into strict 2D WKT notation.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaParseEWKT](#) (const unsigned char *in_buffer)
Creates a Geometry object from EWKT notation.
- GAIAGEO_DECLARE void [gaiaToEWKT](#) ([gaiaOutBufferPtr](#) out_buf, [gaiaGeomCollPtr](#) geom)
Encodes a Geometry object into EWKT notation.
- GAIAGEO_DECLARE void [gaiaOutPointZ](#) ([gaiaOutBufferPtr](#) out_buf, [gaiaPointPtr](#) point)
Encodes a WKT 3D Point [XYZ].
- GAIAGEO_DECLARE void [gaiaOutPointZex](#) ([gaiaOutBufferPtr](#) out_buf, [gaiaPointPtr](#) point, int precision)
Encodes a WKT 3D Point [XYZ].

- GAIAGEO_DECLARE void [gaiaOutLinestringZ](#) ([gaiaOutBufferPtr](#) out_buf, [gaiaLinestringPtr](#) linestring)
Encodes a WKT 3D Linestring [XYZ].
- GAIAGEO_DECLARE void [gaiaOutLinestringZex](#) ([gaiaOutBufferPtr](#) out_buf, [gaiaLinestringPtr](#) linestring, int precision)
Encodes a WKT 3D Linestring [XYZ].
- GAIAGEO_DECLARE void [gaiaOutPolygonZ](#) ([gaiaOutBufferPtr](#) out_buf, [gaiaPolygonPtr](#) polygon)
Encodes a WKT 3D Polygon [XYZ].
- GAIAGEO_DECLARE void [gaiaOutPolygonZex](#) ([gaiaOutBufferPtr](#) out_buf, [gaiaPolygonPtr](#) polygon, int precision)
Encodes a WKT 3D Polygon [XYZ].
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaParseKml](#) (const unsigned char *in_buffer)
Creates a Geometry object from KML notation.
- GAIAGEO_DECLARE void [gaiaOutBareKml](#) ([gaiaOutBufferPtr](#) out_buf, [gaiaGeomCollPtr](#) geom, int precision)
Encodes a Geometry object into KML notation.
- GAIAGEO_DECLARE void [gaiaOutFullKml](#) ([gaiaOutBufferPtr](#) out_buf, const char *name, const char *desc, [gaiaGeomCollPtr](#) geom, int precision)
Encodes a Geometry object into KML notation.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaParseGml](#) (const unsigned char *in_buffer, sqlite3 *sqlite_handle)
Creates a Geometry object from GML notation.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaParseGml_r](#) (const void *p_cache, const unsigned char *in_buffer, sqlite3 *sqlite_handle)
Creates a Geometry object from GML notation.
- GAIAGEO_DECLARE void [gaiaOutGml_ex](#) ([gaiaOutBufferPtr](#) out_buf, int version, int flipped, int precision, [gaiaGeomCollPtr](#) geom)
Encodes a Geometry object into GML notation.
- GAIAGEO_DECLARE void [gaiaOutGml](#) ([gaiaOutBufferPtr](#) out_buf, int version, int precision, [gaiaGeomCollPtr](#) geom)
Encodes a Geometry object into GML notation.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaParseGeoJSON](#) (const unsigned char *in_buffer)
Creates a Geometry object from GeoJSON notation.
- GAIAGEO_DECLARE void [gaiaOutGeoJSON](#) ([gaiaOutBufferPtr](#) out_buf, [gaiaGeomCollPtr](#) geom, int precision, int options)
Encodes a Geometry object into GeoJSON notation.
- GAIAGEO_DECLARE void [gaiaOutSvg](#) ([gaiaOutBufferPtr](#) out_buf, [gaiaGeomCollPtr](#) geom, int relative, int precision)
Encodes a Geometry object into SVG notation.
- GAIAGEO_DECLARE [gaiaValuePtr](#) [gaiaCloneValue](#) ([gaiaValuePtr](#) org)
Allocates a new DBF Field Value object [duplicating an existing one].
- GAIAGEO_DECLARE void [gaiaFreeValue](#) ([gaiaValuePtr](#) p)
Resets a DBF Field Value object to its initial empty state.
- GAIAGEO_DECLARE [gaiaDbfFieldPtr](#) [gaiaAllocDbfField](#) (char *name, unsigned char type, int offset, unsigned char length, unsigned char decimals)
Allocates a new DBF Field object.
- GAIAGEO_DECLARE void [gaiaFreeDbfField](#) ([gaiaDbfFieldPtr](#) p)
Destroys a DBF Field object.
- GAIAGEO_DECLARE [gaiaDbfFieldPtr](#) [gaiaCloneDbfField](#) ([gaiaDbfFieldPtr](#) org)
Allocates a new DBF Field object [duplicating an existing one].
- GAIAGEO_DECLARE void [gaiaSetNullValue](#) ([gaiaDbfFieldPtr](#) field)
Sets a NULL current value for a DBF Field object.
- GAIAGEO_DECLARE void [gaiaSetIntValue](#) ([gaiaDbfFieldPtr](#) field, sqlite3_int64 value)

- Sets an INTEGER current value for a DBF Field object.*

 - GAIAGEO_DECLARE void [gaiaSetDoubleValue](#) ([gaiaDbfFieldPtr](#) field, double value)

Sets a DOUBLE current value for a DBF Field object.
- GAIAGEO_DECLARE void [gaiaSetStrValue](#) ([gaiaDbfFieldPtr](#) field, char *str)

Sets a TEXT current value for a DBF Field object.
- GAIAGEO_DECLARE [gaiaDbfListPtr](#) [gaiaAllocDbfList](#) (void)

Creates an initially empty DBF List object.
- GAIAGEO_DECLARE void [gaiaFreeDbfList](#) ([gaiaDbfListPtr](#) list)

Destroys a DBF List object.
- GAIAGEO_DECLARE int [gaiaIsValidDbfList](#) ([gaiaDbfListPtr](#) list)

Checks a DBF List object for validity.
- GAIAGEO_DECLARE [gaiaDbfFieldPtr](#) [gaiaAddDbfField](#) ([gaiaDbfListPtr](#) list, char *name, unsigned char type, int offset, unsigned char length, unsigned char decimals)

Inserts a further DBF Field object into a DBF List object.
- GAIAGEO_DECLARE void [gaiaResetDbfEntity](#) ([gaiaDbfListPtr](#) list)

Resets a DBF List object to its initial empty state.
- GAIAGEO_DECLARE [gaiaDbfListPtr](#) [gaiaCloneDbfEntity](#) ([gaiaDbfListPtr](#) org)

Allocates a new DBF List object [duplicating an existing one].
- GAIAGEO_DECLARE [gaiaShapefilePtr](#) [gaiaAllocShapefile](#) (void)

Allocates a new Shapefile object.
- GAIAGEO_DECLARE void [gaiaFreeShapefile](#) ([gaiaShapefilePtr](#) shp)

Destroys a Shapefile object.
- GAIAGEO_DECLARE void [gaiaOpenShpRead](#) ([gaiaShapefilePtr](#) shp, const char *path, const char *char↔From, const char *charTo)

Open a Shapefile in read mode.
- GAIAGEO_DECLARE void [gaiaOpenShpWriteEx](#) ([gaiaShapefilePtr](#) shp, const char *path, int shape, [gaiaDbfListPtr](#) list, const char *charFrom, const char *charTo, int colname_case)

Open a Shapefile in write mode - extended.
- GAIAGEO_DECLARE void [gaiaOpenShpWrite](#) ([gaiaShapefilePtr](#) shp, const char *path, int shape, [gaiaDbfListPtr](#) list, const char *charFrom, const char *charTo)

Open a Shapefile in write mode.
- GAIAGEO_DECLARE int [gaiaReadShpEntity](#) ([gaiaShapefilePtr](#) shp, int current_row, int srid)

Reads a feature from a Shapefile object.
- GAIAGEO_DECLARE int [gaiaReadShpEntity_ex](#) ([gaiaShapefilePtr](#) shp, int current_row, int srid, int text_↔dates)

Reads a feature from a Shapefile object.
- GAIAGEO_DECLARE void [gaiaShpAnalyze](#) ([gaiaShapefilePtr](#) shp)

Prescans a Shapefile object gathering informations.
- GAIAGEO_DECLARE int [gaiaWriteShpEntity](#) ([gaiaShapefilePtr](#) shp, [gaiaDbfListPtr](#) entity)

Writes a feature into a Shapefile object.
- GAIAGEO_DECLARE void [gaiaFlushShpHeaders](#) ([gaiaShapefilePtr](#) shp)

Writes into an output Shapefile any required header / footer.
- GAIAGEO_DECLARE [gaiaDbfPtr](#) [gaiaAllocDbf](#) (void)

Allocates a new DBF File object.
- GAIAGEO_DECLARE void [gaiaFreeDbf](#) ([gaiaDbfPtr](#) dbf)

Destroys a DBF File object.
- GAIAGEO_DECLARE void [gaiaOpenDbfRead](#) ([gaiaDbfPtr](#) dbf, const char *path, const char *charFrom, const char *charTo)

Open a DBF File in read mode.
- GAIAGEO_DECLARE [gaiaDbfPtr](#) [gaiaOpenZipDbf](#) (const char *zip_path, const char *filename, const char *charFrom, const char *charTo)

Open a DBF File contained within a Zipfile (just for checking its fields)

- GAIAGEO_DECLARE void [gaiaOpenDbfWriteEx](#) ([gaiaDbfPtr](#) dbf, const char *path, const char *charFrom, const char *charTo, int colname_case)
Open a DBF File in write mode,- extended.
- GAIAGEO_DECLARE void [gaiaOpenDbfWrite](#) ([gaiaDbfPtr](#) dbf, const char *path, const char *charFrom, const char *charTo)
Open a DBF File in write mode.
- GAIAGEO_DECLARE int [gaiaReadDbfEntity](#) ([gaiaDbfPtr](#) dbf, int current_row, int *deleted)
Reads a record from a DBF File object.
- GAIAGEO_DECLARE int [gaiaReadDbfEntity_ex](#) ([gaiaDbfPtr](#) dbf, int current_row, int *deleted, int text_dates)
Reads a record from a DBF File object.
- GAIAGEO_DECLARE int [gaiaWriteDbfEntity](#) ([gaiaDbfPtr](#) dbf, [gaiaDbfListPtr](#) entity)
Writes a record into a DBF File object.
- GAIAGEO_DECLARE void [gaiaFlushDbfHeader](#) ([gaiaDbfPtr](#) dbf)
Writes into an output DBF File any required header / footer.
- GAIAGEO_DECLARE size_t [gaiaMemRead](#) (void *ptr, size_t bytes, [gaiaMemFilePtr](#) mem)
Reads from a Memory File.
- GAIAGEO_DECLARE int [gaiaMemFseek](#) ([gaiaMemFilePtr](#) mem, off_t offset)
Repositioning a Memory File.
- GAIAGEO_DECLARE char * [gaiaReadWktFromZipShp](#) (const char *zip_path, const char *basename)
Attempting to get a WKT from the .PRJ member of a given zipped Shapefile.
- GAIAGEO_DECLARE int [gaiaZipfileNumSHP](#) (const char *zip_path, int *count)
Will return the number of Shapefiles from within a given Zipfile.
- GAIAGEO_DECLARE char * [gaiaZipfileShpN](#) (const char *zip_path, int idx)
Will return the basename of the Nth Shapefile from within a given Zipfile.
- GAIAGEO_DECLARE int [gaiaZipfileNumDBF](#) (const char *zip_path, int *count)
Will return the number of DBF files from within a given Zipfile.
- GAIAGEO_DECLARE char * [gaiaZipfileDbfN](#) (const char *zip_path, int idx)
Will return the filename of the Nth DBF file from within a given Zipfile.
- GAIAGEO_DECLARE [gaiaTextReaderPtr](#) [gaiaTextReaderAlloc](#) (const char *path, char field_separator, char text_separator, char decimal_separator, int first_line_titles, const char *encoding)
Creates a Text Reader object.
- GAIAGEO_DECLARE void [gaiaTextReaderDestroy](#) ([gaiaTextReaderPtr](#) reader)
Destroys a Text Reader object.
- GAIAGEO_DECLARE int [gaiaTextReaderParse](#) ([gaiaTextReaderPtr](#) reader)
Prescans the external file associated to a Text Reade object.
- GAIAGEO_DECLARE int [gaiaTextReaderGetRow](#) ([gaiaTextReaderPtr](#) reader, int row_num)
Reads a line from a Text Reader object.
- GAIAGEO_DECLARE int [gaiaTextReaderFetchField](#) ([gaiaTextReaderPtr](#) reader, int field_num, int *type, const char **value)
Retrieves an individual field value from the current Line.

6.10.1 Detailed Description

Geometry handling functions: formats.

6.10.2 Function Documentation

6.10.2.1 `gaiaAddDbfField()`

```
GAIA GEO_DECLARE gaiaDbfFieldPtr gaiaAddDbfField (
    gaiaDbfListPtr list,
    char * name,
    unsigned char type,
    int offset,
    unsigned char length,
    unsigned char decimals )
```

Inserts a further DBF Field object into a DBF List object.

Parameters

<i>list</i>	pointer to the DBF List object.
<i>name</i>	text string: DBF Field name.
<i>type</i>	identifier of the corresponding DBF data type.
<i>offset</i>	corresponding offset into the DBF I/O buffer.
<i>length</i>	max field length (in bytes).
<i>decimals</i>	precision: number of decimal digits.

Returns

the pointer to newly created DBF Field object.

See also

[gaiaAllocDbfField](#)

Note

supported DBF data types are:

- 'C' text string [default]
- 'N' numeric
- 'D' date
- 'L' boolean

6.10.2.2 `gaiaAllocDbf()`

```
GAIA GEO_DECLARE gaiaDbfPtr gaiaAllocDbf (
    void )
```

Allocates a new DBF File object.

Returns

the pointer to newly created DBF File object.

See also

[gaiaFreeDbf](#), [gaiaOpenDbfRead](#), [gaiaOpenDbfWrite](#), [gaiaReadDbfEntity](#), [gaiaWriteDbfEntity](#), [gaiaFlushDbfHeader](#)

Note

you are responsible to destroy (before or after) any allocated DBF File.
you should phisically open the DBF File in *read* or *write* mode before performing any actual I/O operation.

6.10.2.3 `gaiaAllocDbfField()`

```
GAIAGEO_DECLARE gaiaDbfFieldPtr gaiaAllocDbfField (
    char * name,
    unsigned char type,
    int offset,
    unsigned char length,
    unsigned char decimals )
```

Allocates a new DBF Field object.

Parameters

<i>name</i>	text string: DBF Field name.
<i>type</i>	identifier of the corresponding DBF data type.
<i>offset</i>	corresponding offset into the DBF I/O buffer.
<i>length</i>	max field length (in bytes).
<i>decimals</i>	precision: number of decimal digits.

Returns

the pointer to newly created DBF Field object.

See also

[gaiaFreeDbfField](#), [gaiaCloneDbfField](#), [gaiaFreeValue](#), [gaiaSetNullValue](#), [gaiaSetIntValue](#), [gaiaSetDoubleValue](#), [gaiaSetStrValue](#)

Note

you are responsible to destroy (before or after) any allocated DBF Field, unless you've passed ownership to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

supported DBF data types are:

- 'C' text string [default]
- 'N' numeric
- 'D' date
- 'L' boolean

6.10.2.4 `gaiaAllocDbfList()`

```
GAIA GEO_DECLARE gaiaDbfListPtr gaiaAllocDbfList (
    void )
```

Creates an initially empty DBF List object.

Returns

the pointer to newly allocated DBF List object: NULL on failure.

See also

[gaiaFreeDbfList](#), [gaiaIsValidDbfList](#), [gaiaResetDbfEntity](#), [gaiaCloneDbfEntity](#), [gaiaAddDbfField](#)

Note

you are responsible to destroy (before or after) any allocated DBF List, unless you've passed ownership to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.10.2.5 `gaiaAllocShapefile()`

```
GAIA GEO_DECLARE gaiaShapefilePtr gaiaAllocShapefile (
    void )
```

Allocates a new Shapefile object.

Returns

the pointer to newly created Shapefile object.

See also

[gaiaFreeShapefile](#), [gaiaOpenShpRead](#), [gaiaOpenShpWrite](#), [gaiaReadShpEntity](#), [gaiaShpAnalyze](#), [gaiaWriteShpEntity](#), [gaiaFlushShpHeaders](#)

Note

you are responsible to destroy (before or after) any allocated Shapefile.
you should physically open the Shapefile in *read* or *write* mode before performing any actual I/O operation.

6.10.2.6 `gaiaAppendToOutBuffer()`

```
GAIA GEO_DECLARE void gaiaAppendToOutBuffer (
    gaiaOutBufferPtr buf,
    const char * text )
```

Appends a text string at the end of Text output buffer.

Parameters

<i>buf</i>	pointer to gaiaOutBufferStruct structure.
<i>text</i>	the text string to be appended.

See also

[gaiaOutBufferInitialize](#), [gaiaOutBufferReset](#)

Note

You are required to initialize this structure before attempting any further operation: the dynamically growing Text buffer will be automatically allocated and/or extended as required.

6.10.2.7 [gaiaCloneDbfEntity\(\)](#)

```
GAIA GEO_DECLARE gaiaDbfListPtr gaiaCloneDbfEntity (  
    gaiaDbfListPtr org )
```

Allocates a new DBF List object [duplicating an existing one].

Parameters

<i>org</i>	pointer to input DBF List object.
------------	-----------------------------------

Returns

the pointer to newly created DBF List object.

See also

[gaiaCloneDbfField](#), [gaiaCloneValue](#),

Note

the newly created object is an exact copy of the original one.
this including any currently set Field Value.

6.10.2.8 [gaiaCloneDbfField\(\)](#)

```
GAIA GEO_DECLARE gaiaDbfFieldPtr gaiaCloneDbfField (  
    gaiaDbfFieldPtr org )
```

Allocates a new DBF Field object [duplicating an existing one].

Parameters

<i>org</i>	pointer to input DBF Field object.
------------	------------------------------------

Returns

the pointer to newly created DBF Field object.

See also

[gaiaAllocDbfField](#), [gaiaFreeDbfField](#), [gaiaCloneDbfField](#), [gaiaFreeValue](#), [gaiaSetNullValue](#), [gaiaSetIntValue](#), [gaiaSetDoubleValue](#), [gaiaSetStrValue](#)

Note

the newly created object is an exact copy of the original one [this including an eventual Field Value].

6.10.2.9 gaiaCloneValue()

```
GAIAGEO_DECLARE gaiaValuePtr gaiaCloneValue (
    gaiaValuePtr org )
```

Allocates a new DBF Field Value object [duplicating an existing one].

Parameters

<i>org</i>	pointer to input DBF Field Value object.
------------	--

Returns

the pointer to newly created DBF Field object.

See also

[gaiaAllocDbfField](#), [gaiaFreeDbfField](#), [gaiaCloneDbfField](#), [gaiaCloneValue](#), [gaiaSetNullValue](#), [gaiaSetIntValue](#), [gaiaSetDoubleValue](#), [gaiaSetStrValue](#)

Note

the newly created object is an exact copy of the original one.

6.10.2.10 `gaiaEndianArch()`

```
GAIA GEO_DECLARE int gaiaEndianArch (
    void )
```

Test CPU endianness.

Returns

0 if big-endian: any other value if little-endian

6.10.2.11 `gaiaEwkbGetLinestring()`

```
GAIA GEO_DECLARE int gaiaEwkbGetLinestring (
    gaiaGeomCollPtr geom,
    unsigned char * blob,
    int offset,
    int blob_size,
    int endian,
    int endian_arch,
    int endian_arch,
    int dims )
```

Attempts to decode a Point from within an EWKB binary buffer.

Parameters

<i>geom</i>	pointer to an existing Geometry object; if succesfull the parsed Linestring will be inserted into this Geometry
<i>blob</i>	pointer to EWKB input buffer
<i>offset</i>	the offset (in bytes) on the input buffer where the Point definition is expected to start.
<i>blob_size</i>	length (in bytes) of the input buffer.
<i>endian</i>	(boolean) states if the EWKB input buffer is little- or big-endian encode.
<i>endian_arch</i>	(boolean) states if the target CPU has a little- or big-endian architecture.
<i>dims</i>	dimensions: one of GAIA_XY, GAIA_XY_Z, GAIA_XY_M or GAIA_XY_Z_M

Returns

-1 on failure; otherwise the offset where the next object starts.

See also

[gaiaEwkbGetPoint](#), [gaiaEwkbGetPolygon](#), [gaiaEwkbGetMultiGeometry](#)

Note

these functions are mainly intended for internal usage.

6.10.2.12 `gaiaEwkbGetMultiGeometry()`

```
GAIA_GEO_DECLARE int gaiaEwkbGetMultiGeometry (
    gaiaGeomCollPtr geom,
    unsigned char * blob,
    int offset,
    int blob_size,
    int endian,
    int endian_arch,
    int dims )
```

Attempts to decode a MultiGeometry from within an EWKB binary buffer.

Parameters

<i>geom</i>	pointer to an existing Geometry object; if successful the parsed MultiGeometry will be inserted into this Geometry
<i>blob</i>	pointer to EWKB input buffer
<i>offset</i>	the offset (in bytes) on the input buffer where the Point definition is expected to start.
<i>blob_size</i>	length (in bytes) of the input buffer.
<i>endian</i>	(boolean) states if the EWKB input buffer is little- or big-endian encode.
<i>endian_arch</i>	(boolean) states if the target CPU has a little- or big-endian architecture.
<i>dims</i>	dimensions: one of GAIA_XY, GAIA_XY_Z, GAIA_XY_M or GAIA_XY_Z_M

Returns

-1 on failure; otherwise the offset where the next object starts.

See also

[gaiaEwkbGetPoint](#), [gaiaEwkbGetLinestring](#), [gaiaEwkbGetPolygon](#)

Note

these functions are mainly intended for internal usage.

6.10.2.13 `gaiaEwkbGetPoint()`

```
GAIA_GEO_DECLARE int gaiaEwkbGetPoint (
    gaiaGeomCollPtr geom,
    unsigned char * blob,
    int offset,
    int blob_size,
    int endian,
    int endian_arch,
    int dims )
```

Attempts to decode a Point from within an EWKB binary buffer.

Parameters

<i>geom</i>	pointer to an existing Geometry object; if succesfull the parsed Point will be inserted into this Geometry
<i>blob</i>	pointer to EWKB input buffer
<i>offset</i>	the offset (in bytes) on the input buffer where the Point definition is expected to start.
<i>blob_size</i>	length (in bytes) of the input buffer.
<i>endian</i>	(boolean) states if the EWKB input buffer is little- or big-endian encode.
<i>endian_arch</i>	(boolean) states if the target CPU has a little- or big-endian architecture.
<i>dims</i>	dimensions: one of GAIA_XY, GAIA_XY_Z, GAIA_XY_M or GAIA_XY_Z_M

Returns

-1 on failure; otherwise the offset where the next object starts.

See also

[gaiaEwkbGetLinestring](#), [gaiaEwkbGetPolygon](#), [gaiaEwkbGetMultiGeometry](#)

Note

these functions are mainly intended for internal usage.

6.10.2.14 **gaiaEwkbGetPolygon()**

```

GAIA_GEO_DECLARE int gaiaEwkbGetPolygon (
    gaiaGeomCollPtr geom,
    unsigned char * blob,
    int offset,
    int blob_size,
    int endian,
    int endian_arch,
    int dims )

```

Attempts to decode a Polygon from within an EWKB binary buffer.

Parameters

<i>geom</i>	pointer to an existing Geometry object; if succesfull the parsed Polygon will be inserted into this Geometry
<i>blob</i>	pointer to EWKB input buffer
<i>offset</i>	the offset (in bytes) on the input buffer where the Point definition is expected to start.
<i>blob_size</i>	length (in bytes) of the input buffer.
<i>endian</i>	(boolean) states if the EWKB input buffer is little- or big-endian encode.
<i>endian_arch</i>	(boolean) states if the target CPU has a little- or big-endian architecture.
<i>dims</i>	dimensions: one of GAIA_XY, GAIA_XY_Z, GAIA_XY_M or GAIA_XY_Z_M

Returns

-1 on failure; otherwise the offset where the next object starts.

See also

[gaiaEwkbGetPoint](#), [gaiaEwkbGetPolygon](#), [gaiaEwkbGetMultiGeometry](#)

6.10.2.15 gaiaExport16()

```
GAIAGEO_DECLARE void gaiaExport16 (
    unsigned char * p,
    short value,
    int little_endian,
    int little_endian_arch )
```

Export an INT-16 value in endian-aware fashion.

Parameters

<i>p</i>	endian-dependent representation (output buffer).
<i>value</i>	the internal value to be exported.
<i>little_endian</i>	0 if the output buffer has to be big-endian: any other value for little-endian.
<i>little_endian_arch</i>	the value returned by gaiaEndianArch()

See also

[gaiaEndianArch](#), [gaiaImport16](#)

Note

you are expected to pass an output buffer corresponding to an allocation size of (at least) 2 bytes.

6.10.2.16 gaiaExport32()

```
GAIAGEO_DECLARE void gaiaExport32 (
    unsigned char * p,
    int value,
    int little_endian,
    int little_endian_arch )
```

Export an INT-32 value in endian-aware fashion.

Parameters

<i>p</i>	endian-dependent representation (output buffer).
<i>value</i>	the internal value to be exported.
<i>little_endian</i>	0 if the output buffer has to be big-endian: any other value for little-endian.
<i>little_endian_arch</i>	the value returned by gaiaEndianArch()

See also

[gaiaEndianArch](#), [gaiaImport32](#)

Note

you are expected to pass an output buffer corresponding to an allocation size of (at least) 4 bytes.

6.10.2.17 `gaiaExport64()`

```
GAIAGEO_DECLARE void gaiaExport64 (
    unsigned char * p,
    double value,
    int little_endian,
    int little_endian_arch )
```

Export a DOUBLE value in endian-aware fashion.

Parameters

<i>p</i>	endian-dependent representation (output buffer).
<i>value</i>	the internal value to be exported.
<i>little_endian</i>	0 if the output buffer has to be big-endian: any other value for little-endian.
<i>little_endian_arch</i>	the value returned by gaiaEndianArch()

See also

[gaiaEndianArch](#), [gaiaImport64](#)

Note

you are expected to pass an output buffer corresponding to an allocation size of (at least) 8 bytes.

6.10.2.18 `gaiaExportF32()`

```
GAIAGEO_DECLARE void gaiaExportF32 (
    unsigned char * p,
    float value,
    int little_endian,
    int little_endian_arch )
```

Export a FLOAT-32 value in endian-aware fashion.

Parameters

<i>p</i>	endian-dependent representation (output buffer).
<i>value</i>	the internal value to be exported.
<i>little_endian</i>	0 if the output buffer has to be big-endian: any other value for little-endian.
<i>little_endian_arch</i>	the value returned by gaiaEndianArch()

See also

[gaiaEndianArch](#), [gaiaImportF32](#)

Note

you are expected to pass an output buffer corresponding to an allocation size of (at least) 4 bytes.

6.10.2.19 `gaiaExportI64()`

```
GAIAGEO_DECLARE void gaiaExportI64 (
    unsigned char * p,
    sqlite3_int64 value,
    int little_endian,
    int little_endian_arch )
```

Export an INT-64 value in endian-aware fashion.

Parameters

<i>p</i>	endian-dependent representation (output buffer).
<i>value</i>	the internal value to be exported.
<i>little_endian</i>	0 if the output buffer has to be big-endian: any other value for little-endian.
<i>little_endian_arch</i>	the value returned by gaiaEndianArch()

See also

[gaiaEndianArch](#), [gaiaImportI64](#)

Note

you are expected to pass an output buffer corresponding to an allocation size of (at least) 8 bytes.

6.10.2.20 `gaiaExportU32()`

```
GAIAGEO_DECLARE void gaiaExportU32 (
    unsigned char * p,
    unsigned int value,
    int little_endian,
    int little_endian_arch )
```

Export an UINT-32 value in endian-aware fashion.

Parameters

<i>p</i>	endian-dependent representation (output buffer).
<i>value</i>	the internal value to be exported.
<i>little_endian</i>	0 if the output buffer has to be big-endian: any other value for little-endian.
<i>little_endian_arch</i>	the value returned by gaiaEndianArch()

See also

[gaiaEndianArch](#), [gaiaImportU32](#)

Note

you are expected to pass an output buffer corresponding to an allocation size of (at least) 4 bytes.

6.10.2.21 `gaiaFlushDbfHeader()`

```
GAIA GEO_DECLARE void gaiaFlushDbfHeader (
    gaiaDbfPtr dbf )
```

Writes into an output DBF File any required header / footer.

Parameters

<i>dbf</i>	pointer to the DBF File object.
------------	---------------------------------

See also

[gaiaAllocDbf](#), [gaiaFreeDbf](#), [gaiaOpenDbfRead](#), [gaiaOpenDbfWrite](#), [gaiaReadDbfEntity](#), [gaiaWriteDbfEntity](#)

Note

forgetting to call `gaiaFlushDbfHeader` for any DBF File opened in *write* mode immediately before destroying the object, will surely cause severe file corruption.

6.10.2.22 `gaiaFlushShpHeaders()`

```
GAIA GEO_DECLARE void gaiaFlushShpHeaders (
    gaiaShapefilePtr shp )
```

Writes into an output Shapefile any required header / footer.

Parameters

<i>shp</i>	pointer to the Shapefile object.
------------	----------------------------------

See also

[gaiaAllocShapefile](#), [gaiaFreeShapefile](#), [gaiaOpenShpRead](#), [gaiaOpenShpWrite](#), [gaiaReadShpEntity](#), [gaiaShpAnalyze](#), [gaiaWriteShpEntity](#)

Note

forgetting to call `gaiaFlushShpHeader` for any Shapefile opened in *write* mode immediately before destroying the object, will surely cause severe file corruption.

6.10.2.23 `gaiaFreeDbf()`

```
GAIAGEO_DECLARE void gaiaFreeDbf (
    gaiaDbfPtr dbf )
```

Destroys a DBF File object.

Parameters

<i>dbf</i>	pointer to the DBF File object.
------------	---------------------------------

See also

[gaiaAllocDbf](#), [gaiaFreeDbf](#), [gaiaOpenDbfWrite](#), [gaiaReadDbfEntity](#), [gaiaWriteDbfEntity](#), [gaiaFlushDbfHeader](#)

Note

destroying the Shapefile object will close any related file: anyway you are responsible to explicitly call `gaiaFlushShpHeader` before destroying a Shapefile opened in *write* mode.

6.10.2.24 `gaiaFreeDbfField()`

```
GAIAGEO_DECLARE void gaiaFreeDbfField (
    gaiaDbfFieldPtr p )
```

Destroys a DBF Field object.

Parameters

<i>p</i>	pointer to DBF Field object
----------	-----------------------------

See also

[gaiaAllocDbfField](#), [gaiaCloneDbfField](#), [gaiaCloneValue](#), [gaiaFreeValue](#), [gaiaSetNullValue](#), [gaiaSetIntValue](#), [gaiaSetDoubleValue](#), [gaiaSetStrValue](#)

6.10.2.25 `gaiaFreeDbfList()`

```
GAIA GEO_DECLARE void gaiaFreeDbfList (
    gaiaDbfListPtr list )
```

Destroys a DBF List object.

Parameters

<i>list</i>	pointer to the DBF List object
-------------	--------------------------------

See also

[gaiaAllocDbfList](#), [gaiaIsValidDbfList](#), [gaiaResetDbfEntity](#), [gaiaCloneDbfEntity](#), [gaiaAddDbfField](#)

Note

attempting to destroy any DBF List object whose ownership has already been transferred to some other (higher order) object is a serious error, and will easily cause severe memory corruption.

6.10.2.26 `gaiaFreeShapefile()`

```
GAIA GEO_DECLARE void gaiaFreeShapefile (
    gaiaShapefilePtr shp )
```

Destroys a Shapefile object.

Parameters

<i>shp</i>	pointer to the Shapefile object.
------------	----------------------------------

See also

[gaiaAllocShapefile](#), [gaiaOpenShpRead](#), [gaiaOpenShpWrite](#), [gaiaReadShpEntity](#), [gaiaShpAnalyze](#), [gaiaWriteShpEntity](#), [gaiaFlushShpHeaders](#)

Note

destroying the Shapefile object will close any related file: anyway you are responsible to explicitly call `gaiaFlushShpHeader` before destroying a Shapefile opened in *write* mode.

6.10.2.27 `gaiaFreeValue()`

```
GAIA GEO_DECLARE void gaiaFreeValue (
    gaiaValuePtr p )
```

Resets a DBF Field Value object to its initial empty state.

Parameters

<i>p</i>	pointer to DBF Field Value object
----------	-----------------------------------

See also

[gaiaAllocDbfField](#), [gaiaCloneDbfField](#), [gaiaCloneValue](#), [gaiaSetNullValue](#), [gaiaSetIntValue](#), [gaiaSetDoubleValue](#), [gaiaSetStrValue](#), [gaiaResetDbfEntity](#)

6.10.2.28 `gaiaFromEWKB()`

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaFromEWKB (
    const unsigned char * in_buffer )
```

Creates a Geometry object from EWKB notation.

Parameters

<i>in_buffer</i>	pointer to EWKB buffer
------------------	------------------------

Returns

the pointer to the newly created Geometry object: NULL on failure.

See also

[gaiaToWkb](#), [gaiaToHexWkb](#), [gaiaParseHexEWKB](#), [gaiaToEWKB](#), [gaiaEwkbGetPoint](#), [gaiaEwkbGetLinestring](#), [gaiaEwkbGetPolygon](#), [gaiaEwkbGetMultiGeometry](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, unless you've passed ownership of the Geometry object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.10.2.29 `gaiaFromFgf()`

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaFromFgf (
    const unsigned char * blob,
    unsigned int size )
```

Creates a Geometry object from FGF notation.

Parameters

<i>blob</i>	pointer to FGF buffer
<i>size</i>	the BLOB's size (in bytes)

Returns

the pointer to the newly created Geometry object: NULL on failure.

See also

[gaiaToFgf](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, unless you've passed ownership of the Geometry object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.10.2.30 gaiaFromSpatialiteBlobWkb()

```
GAIA_GEO_DECLARE gaiaGeomCollPtr gaiaFromSpatialiteBlobWkb (
    const unsigned char * blob,
    unsigned int size )
```

Creates a Geometry object from the corresponding BLOB-Geometry.

Parameters

<i>blob</i>	pointer to BLOB-Geometry
<i>size</i>	the BLOB's size

Returns

the pointer to the newly created Geometry object: NULL on failure

See also

[gaiaFreeGeomColl](#), [gaiaToSpatialiteBlobWkb](#), [gaiaToCompressedBlobWkb](#), [gaiaFromSpatialiteBlobWkbEx](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, unless you've passed ownership of the Geometry object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

Examples

[demo1.c](#).

6.10.2.31 `gaiaFromSpatialLiteBlobWkbEx()`

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaFromSpatialLiteBlobWkbEx (
    const unsigned char * blob,
    unsigned int size,
    int gpkg_mode,
    int gpkg_amphibious )
```

Creates a Geometry object from the corresponding BLOB-Geometry.

Parameters

<i>blob</i>	pointer to BLOB-Geometry
<i>size</i>	the BLOB's size
<i>gpkg_mode</i>	is set to TRUE will accept only GPKG Geometry-BLOBs
<i>gpkg_amphibious</i>	is set to TRUE will indifferently accept either SpatialLite Geometry-BLOBs or GPKG Geometry-BLOBs

Returns

the pointer to the newly created Geometry object: NULL on failure

See also

[gaiaFreeGeomColl](#), [gaiaToSpatialLiteBlobWkb](#), [gaiaToCompressedBlobWkb](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, unless you've passed ownership of the Geometry object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.10.2.32 `gaiaFromWkb()`

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaFromWkb (
    const unsigned char * blob,
    unsigned int size )
```

Creates a Geometry object from WKB notation.

Parameters

<i>blob</i>	pointer to WKB buffer
<i>size</i>	the BLOB's size (in bytes)

Returns

the pointer to the newly created Geometry object: NULL on failure.

See also

[gaiaToWkb](#), [gaiaToHexWkb](#), [gaiaFromEWKB](#), [gaiaToEWKB](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, unless you've passed ownership of the Geometry object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.10.2.33 gaiaImport16()

```
GAIA GEO_DECLARE short gaiaImport16 (
    const unsigned char * p,
    int little_endian,
    int little_endian_arch )
```

Import an INT-16 value in endian-aware fashion.

Parameters

<i>p</i>	endian-dependent representation (input buffer).
<i>little_endian</i>	0 if the input buffer is big-endian: any other value for little-endian.
<i>little_endian_arch</i>	the value returned by gaiaEndianArch()

Returns

the internal SHORT value

See also

[gaiaEndianArch](#), [gaiaExport16](#)

Note

you are expected to pass an input buffer corresponding to an allocation size of (at least) 2 bytes.

6.10.2.34 gaiaImport32()

```
GAIA GEO_DECLARE int gaiaImport32 (
    const unsigned char * p,
    int little_endian,
    int little_endian_arch )
```

Import an INT-32 value in endian-aware fashion.

Parameters

<i>p</i>	endian-dependent representation (input buffer).
<i>little_endian</i>	0 if the input buffer is big-endian: any other value for little-endian.
<i>little_endian_arch</i>	the value returned by gaiaEndianArch()

Returns

the internal INT value

See also

[gaiaEndianArch](#), [gaiaExport32](#)

Note

you are expected to pass an input buffer corresponding to an allocation size of (at least) 4 bytes.

6.10.2.35 gaiaImport64()

```
GAIA GEO_DECLARE double gaiaImport64 (
    const unsigned char * p,
    int little_endian,
    int little_endian_arch )
```

Import an DOUBLE-64 in endian-aware fashion.

Parameters

<i>p</i>	endian-dependent representation (input buffer).
<i>little_endian</i>	0 if the input buffer is big-endian: any other value for little-endian.
<i>little_endian_arch</i>	the value returned by gaiaEndianArch()

Returns

the internal DOUBLE value

See also

[gaiaEndianArch](#), [gaiaExport64](#)

Note

you are expected to pass an input buffer corresponding to an allocation size of (at least) 8 bytes.

6.10.2.36 gaiaImportF32()

```
GAIA GEO_DECLARE float gaiaImportF32 (
    const unsigned char * p,
    int little_endian,
    int little_endian_arch )
```

Import a FLOAT-32 value in endian-aware fashion.

Parameters

<i>p</i>	endian-dependent representation (input buffer).
<i>little_endian</i>	0 if the input buffer is big-endian: any other value for little-endian.
<i>little_endian_arch</i>	the value returned by gaiaEndianArch()

Returns

the internal FLOAT value

See also

[gaiaEndianArch](#), [gaiaExportF32](#)

Note

you are expected to pass an input buffer corresponding to an allocation size of (at least) 4 bytes.

6.10.2.37 gaiaImportI64()

```
GAIA GEO_DECLARE sqlite3_int64 gaiaImportI64 (
    const unsigned char * p,
    int little_endian,
    int little_endian_arch )
```

Import an INT-64 in endian-aware fashion.

Parameters

<i>p</i>	endian-dependent representation (input buffer).
<i>little_endian</i>	0 if the input buffer is big-endian: any other value for little-endian.
<i>little_endian_arch</i>	the value returned by gaiaEndianArch()

Returns

the internal INT-64 value

See also

[gaiaEndianArch](#), [gaiaExportI64](#)

Note

you are expected to pass an input buffer corresponding to an allocation size of (at least) 8 bytes.

6.10.2.38 gaiaImportU32()

```
GAIA GEO_DECLARE unsigned int gaiaImportU32 (
    const unsigned char * p,
    int little_endian,
    int little_endian_arch )
```

Import an UINT-32 value in endian-aware fashion.

Parameters

<i>p</i>	endian-dependent representation (input buffer).
<i>little_endian</i>	0 if the input buffer is big-endian: any other value for little-endian.
<i>little_endian_arch</i>	the value returned by gaiaEndianArch()

Returns

the internal UINT value

See also

[gaiaEndianArch](#), [gaiaExportU32](#)

Note

you are expected to pass an input buffer corresponding to an allocation size of (at least) 4 bytes.

6.10.2.39 gaiaIsValidDbfList()

```
GAIA GEO_DECLARE int gaiaIsValidDbfList (
    gaiaDbfListPtr list )
```

Checks a DBF List object for validity.

Parameters

<i>list</i>	pointer to the DBF List object.
-------------	---------------------------------

Returns

0 if not valid: any other value if valid.

See also

[gaiaAllocDbfList](#), [gaiaFreeDbfList](#), [gaiaIsValidDbfList](#), [gaiaResetDbfEntity](#), [gaiaCloneDbfEntity](#), [gaiaAddDbfField](#)

6.10.2.40 gaiaMakeLine()

```
GAIAGEO_DECLARE void gaiaMakeLine (
    gaiaGeomCollPtr geom1,
    gaiaGeomCollPtr geom2,
    unsigned char ** result,
    int * size )
```

Creates a BLOB-Geometry representing a Segment (2-Points Linestring)

Parameters

<i>geom1</i>	pointer to first Geometry object (expected to represent a Point).
<i>geom2</i>	pointer to second Geometry object (expected to represent a Point).
<i>result</i>	on completion will contain a pointer to BLOB-Geometry: NULL on failure.
<i>size</i>	on completion this variable will contain the BLOB's size (in bytes)

See also

[gaiaFromSpatialiteBlobWkb](#)

Note

the BLOB buffer corresponds to dynamically allocated memory: so you are responsible to free() it [unless SQLite will take care of memory cleanup via buffer binding].

6.10.2.41 gaiaMakePoint()

```
GAIAGEO_DECLARE void gaiaMakePoint (
    double x,
    double y,
    int srid,
    unsigned char ** result,
    int * size )
```

Creates a BLOB-Geometry representing a Point (BLOB-Geometry)

Parameters

<i>x</i>	Point X coordinate.
<i>y</i>	Point Y coordinate.
<i>srid</i>	the SRID to be set for the Point.
<i>result</i>	on completion will contain a pointer to BLOB-Geometry: NULL on failure.
<i>size</i>	on completion this variable will contain the BLOB's size (in bytes)

See also

[gaiaFromSpatialLiteBlobWkb](#), [gaiaMakePointEx](#)

Note

the BLOB buffer corresponds to dynamically allocated memory: so you are responsible to free() it [unless SQLite will take care of memory cleanup via buffer binding].

6.10.2.42 **gaiaMakePointEx()**

```
GAIA_GEO_DECLARE void gaiaMakePointEx (
    int tiny_point,
    double x,
    double y,
    int srid,
    unsigned char ** result,
    int * size )
```

Creates a BLOB-Geometry representing a Point (BLOB-Geometry or BLOB-TinyPoint)

Parameters

<i>tiny_point</i>	if set to TRUE the POINT Geometry will be encoded by using the TinyPoint BLOB format.
<i>x</i>	Point X coordinate.
<i>y</i>	Point Y coordinate.
<i>srid</i>	the SRID to be set for the Point.
<i>result</i>	on completion will contain a pointer to BLOB-Geometry: NULL on failure.
<i>size</i>	on completion this variable will contain the BLOB's size (in bytes)

See also

[gaiaFromSpatialLiteBlobWkb](#), [gaiaMakePoint](#)

Note

the BLOB buffer corresponds to dynamically allocated memory: so you are responsible to free() it [unless SQLite will take care of memory cleanup via buffer binding].

6.10.2.43 gaiaMakePointM()

```
GAIA GEO_DECLARE void gaiaMakePointM (
    double x,
    double y,
    double m,
    int srid,
    unsigned char ** result,
    int * size )
```

Creates a BLOB-Geometry representing a PointM (BLOB-Geometry)

Parameters

<i>x</i>	Point X coordinate.
<i>y</i>	Point Y coordinate.
<i>m</i>	Point M coordinate.
<i>srid</i>	the SRID to be set for the Point.
<i>result</i>	on completion will contain a pointer to BLOB-Geometry: NULL on failure.
<i>size</i>	on completion this variable will contain the BLOB's size (in bytes)

See also

[gaiaFromSpatialiteBlobWkb](#), [gaiaMakePointMEx](#)

Note

the BLOB buffer corresponds to dynamically allocated memory: so you are responsible to free() it [unless SQLite will take care of memory cleanup via buffer binding].

6.10.2.44 gaiaMakePointMEx()

```
GAIA GEO_DECLARE void gaiaMakePointMEx (
    int tiny_point,
    double x,
    double y,
    double m,
    int srid,
    unsigned char ** result,
    int * size )
```

Creates a BLOB-Geometry representing a PointM (BLOB-Geometry or BLOB-TinyPoint)

Parameters

<i>tiny_point</i>	if set to TRUE the POINT Geometry will be encoded by using the TinyPoint BLOB format.
<i>x</i>	Point X coordinate.
<i>y</i>	Point Y coordinate.
<i>m</i>	Point M coordinate.
<i>srid</i>	the SRID to be set for the Point.
<i>result</i>	on completion will contain a pointer to BLOB-Geometry: NULL on failure.
<i>size</i>	on completion this variable will contain the BLOB's size (in bytes)

See also

[gaiaFromSpatialLiteBlobWkb](#), [gaiaMakePointM](#)

Note

the BLOB buffer corresponds to dynamically allocated memory: so you are responsible to free() it [unless SQLite will take care of memory cleanup via buffer binding].

6.10.2.45 `gaiaMakePointZ()`

```
GAIA_GEO_DECLARE void gaiaMakePointZ (
    double x,
    double y,
    double z,
    int srid,
    unsigned char ** result,
    int * size )
```

Creates a BLOB-Geometry representing a PointZ (BLOB-Geometry)

Parameters

<i>x</i>	Point X coordinate.
<i>y</i>	Point Y coordinate.
<i>z</i>	Point Z coordinate.
<i>srid</i>	the SRID to be set for the Point.
<i>result</i>	on completion will contain a pointer to BLOB-Geometry: NULL on failure.
<i>size</i>	on completion this variable will contain the BLOB's size (in bytes)

See also

[gaiaFromSpatialLiteBlobWkb](#), [gaiaMakePointZEx](#)

Note

the BLOB buffer corresponds to dynamically allocated memory: so you are responsible to free() it [unless SQLite will take care of memory cleanup via buffer binding].

6.10.2.46 `gaiaMakePointZEx()`

```
GAIA_GEO_DECLARE void gaiaMakePointZEx (
    int tiny_point,
    double x,
    double y,
    double z,
```

```
int srid,  
unsigned char ** result,  
int * size )
```

Creates a BLOB-Geometry representing a PointZ (BLOB-Geometry or BLOB-TinyPoint)

Parameters

<i>tiny_point</i>	if set to TRUE the POINT Geometry will be encoded by using the TinyPoint BLOB format.
<i>x</i>	Point X coordinate.
<i>y</i>	Point Y coordinate.
<i>z</i>	Point Z coordinate.
<i>srid</i>	the SRID to be set for the Point.
<i>result</i>	on completion will contain a pointer to BLOB-Geometry: NULL on failure.
<i>size</i>	on completion this variable will contain the BLOB's size (in bytes)

See also

[gaiaFromSpatialLiteBlobWkb](#), [gaiaMakePointZ](#)

Note

the BLOB buffer corresponds to dynamically allocated memory: so you are responsible to free() it [unless SQLite will take care of memory cleanup via buffer binding].

6.10.2.47 **gaiaMakePointZM()**

```
GAIA GEO_DECLARE void gaiaMakePointZM (
    double x,
    double y,
    double z,
    double m,
    int srid,
    unsigned char ** result,
    int * size )
```

Creates a BLOB-Geometry representing a PointZM (BLOB-Geometry)

Parameters

<i>x</i>	Point X coordinate.
<i>y</i>	Point Y coordinate.
<i>z</i>	Point Z coordinate.
<i>m</i>	Point M coordinate.
<i>srid</i>	the SRID to be set for the Point.
<i>result</i>	on completion will contain a pointer to BLOB-Geometry: NULL on failure.
<i>size</i>	on completion this variable will contain the BLOB's size (in bytes)

See also

[gaiaFromSpatialLiteBlobWkb](#), [gaiaMakePointZMEx](#)

Note

the BLOB buffer corresponds to dynamically allocated memory: so you are responsible to free() it [unless SQLite will take care of memory cleanup via buffer binding].

6.10.2.48 gaiaMakePointZMEx()

```
GAIA GEO_DECLARE void gaiaMakePointZMEx (
    int tiny_point,
    double x,
    double y,
    double z,
    double m,
    int srid,
    unsigned char ** result,
    int * size )
```

Creates a BLOB-Geometry representing a PointZM (BLOB-Geometry or BLOB-TinyPoint)

Parameters

<i>tiny_point</i>	if set to TRUE the POINT Geometry will be encoded by using the TinyPoint BLOB format.
<i>x</i>	Point X coordinate.
<i>y</i>	Point Y coordinate.
<i>z</i>	Point Z coordinate.
<i>m</i>	Point M coordinate.
<i>srid</i>	the SRID to be set for the Point.
<i>result</i>	on completion will contain a pointer to BLOB-Geometry: NULL on failure.
<i>size</i>	on completion this variable will contain the BLOB's size (in bytes)

See also

[gaiaFromSpatialiteBlobWkb](#), [gaiaMakePointZM](#)

Note

the BLOB buffer corresponds to dynamically allocated memory: so you are responsible to free() it [unless SQLite will take care of memory cleanup via buffer binding].

6.10.2.49 gaiaMemFseek()

```
GAIA GEO_DECLARE int gaiaMemFseek (
    gaiaMemFilePtr mem,
    off_t offset )
```

Repositioning a Memory File.

Parameters

<i>mem</i>	pointer to the Memry File object.
<i>offset</i>	file offset relative to the start of file

Returns

0 on success; -1 on failure.

See also

[gaiaFseek](#)

6.10.2.50 gaiaMemRead()

```
GAIO_GEO_DECLARE size_t gaiaMemRead (
    void * ptr,
    size_t bytes,
    gaiaMemFilePtr mem )
```

Reads from a Memory File.

Parameters

<i>ptr</i>	pointer to the output buffer.
<i>bytes</i>	numeber of bytes to read.
<i>mem</i>	pointer to the Memory File object.

Returns

the number of bytes read.

See also

[gaiaMemFseek](#)

6.10.2.51 gaiaOpenDbfRead()

```
GAIO_GEO_DECLARE void gaiaOpenDbfRead (
    gaiaDbfPtr dbf,
    const char * path,
    const char * charFrom,
    const char * charTo )
```

Open a DBF File in read mode.

Parameters

<i>dbf</i>	pointer to the DBF File object.
<i>path</i>	pathname to the corresponding file-system file.
<i>charFrom</i>	GNU ICONV name identifying the input charset encoding.
<i>charTo</i>	GNU ICONV name identifying the output charset encoding.

See also

[gaiaAllocDbf](#), [gaiaFreeDbf](#), [gaiaOpenDbfWrite](#), [gaiaOpenZipDbf](#), [gaiaReadDbfEntity](#), [gaiaWriteDbfEntity](#), [gaiaFlushDbfHeader](#)

Note

on failure the object member *Valid* will be set to 0; and the object member *LastError* will contain the appropriate error message.

6.10.2.52 [gaiaOpenDbfWrite\(\)](#)

```
GAIA_GEO_DECLARE void gaiaOpenDbfWrite (
    gaiaDbfPtr dbf,
    const char * path,
    const char * charFrom,
    const char * charTo )
```

Open a DBF File in write mode.

Parameters

<i>dbf</i>	pointer to the DBF File object.
<i>path</i>	pathname to the corresponding file-system file.
<i>charFrom</i>	GNU ICONV name identifying the input charset encoding.
<i>charTo</i>	GNU ICONV name identifying the output charset encoding.

See also

[gaiaAllocDbf](#), [gaiaFreeDbf](#), [gaiaOpenDbfRead](#), [gaiaReadDbfEntity](#), [gaiaWriteDbfEntity](#), [gaiaFlushDbfHeader](#), [gaiaOpenDbfWriteEx](#)

Note

simply calls [gaiaOpenDbfWriteEx\(\)](#) by specifying `GAIA_DBF_COLNAME_LOWERCASE`

6.10.2.53 `gaiaOpenDbfWriteEx()`

```
GAIA GEO_DECLARE void gaiaOpenDbfWriteEx (
    gaiaDbfPtr dbf,
    const char * path,
    const char * charFrom,
    const char * charTo,
    int colname_case )
```

Open a DBF File in write mode,- extended.

Parameters

<i>dbf</i>	pointer to the DBF File object.
<i>path</i>	pathname to the corresponding file-system file.
<i>charFrom</i>	GNU ICONV name identifying the input charset encoding.
<i>charTo</i>	GNU ICONV name identifying the output charset encoding.
<i>colname_case</i>	one between GAIA_DBF_COLNAME_LOWERCASE, GAIA_DBF_COLNAME_UPPERCASE or GAIA_DBF_COLNAME_CASE_IGNORE.

See also

[gaiaAllocDbf](#), [gaiaFreeDbf](#), [gaiaOpenDbfRead](#), [gaiaReadDbfEntity](#), [gaiaWriteDbfEntity](#), [gaiaFlushDbfHeader](#), [gaiaOpenDbfWrite](#)

Note

on failure the object member *Valid* will be set to 0; and the object member *LastError* will contain the appropriate error message.

6.10.2.54 `gaiaOpenShpRead()`

```
GAIA GEO_DECLARE void gaiaOpenShpRead (
    gaiaShapefilePtr shp,
    const char * path,
    const char * charFrom,
    const char * charTo )
```

Open a Shapefile in read mode.

Parameters

<i>shp</i>	pointer to the Shapefile object.
<i>path</i>	<i>abstract</i> pathname to the corresponding file-system files.
<i>charFrom</i>	GNU ICONV name identifying the input charset encoding.
<i>charTo</i>	GNU ICONV name identifying the output charset encoding.

See also

[gaiaAllocShapefile](#), [gaiaFreeShapefile](#), [gaiaOpenShpWrite](#), [gaiaReadShpEntity](#), [gaiaShpAnalyze](#), [gaiaWriteShpEntity](#), [gaiaFlushShpHeaders](#)

Note

on failure the object member *Valid* will be set to 0; and the object member *LastError* will contain the appropriate error message.

the *abstract* pathname should not contain any suffix at all.

6.10.2.55 [gaiaOpenShpWrite\(\)](#)

```
GAIA GEO_DECLARE void gaiaOpenShpWrite (
    gaiaShapefilePtr shp,
    const char * path,
    int shape,
    gaiaDbfListPtr list,
    const char * charFrom,
    const char * charTo )
```

Open a Shapefile in write mode.

Parameters

<i>shp</i>	pointer to the Shapefile object.
<i>path</i>	<i>abstract</i> pathname to the corresponding file-system files.
<i>shape</i>	the SHAPE code; expected to be one of GAIA_SHP_POINT, GAIA_SHP_POLYLINE, GAIA_SHP_POLYGON, GAIA_SHP_MULTIPPOINT, GAIA_SHP_POINTZ, GAIA_SHP_POLYLINEZ, GAIA_SHP_POLYGONZ, GAIA_SHP_MULTIPPOINTZ, GAIA_SHP_POINTM, GAIA_SHP_POLYLINEM, GAIA_SHP_POLYGONM, GAIA_SHP_MULTIPPOINTM
<i>list</i>	pointer to DBF List object representing the corresponding data attributes.
<i>charFrom</i>	GNU ICONV name identifying the input charset encoding.
<i>charTo</i>	GNU ICONV name identifying the output charset encoding.

See also

[gaiaAllocShapefile](#), [gaiaFreeShapefile](#), [gaiaOpenShpRead](#), [gaiaReadShpEntity](#), [gaiaShpAnalyze](#), [gaiaWriteShpEntity](#), [gaiaFlushShpHeaders](#), [gaiaOpenShpWriteEx](#)

Note

simply calls [gaiaOpenShpWriteEx\(\)](#) by specifying GAIA_DBF_COLNAME_LOWERCASE

6.10.2.56 `gaiaOpenShpWriteEx()`

```
GAIA GEO_DECLARE void gaiaOpenShpWriteEx (
    gaiaShapefilePtr shp,
    const char * path,
    int shape,
    gaiaDbfListPtr list,
    const char * charFrom,
    const char * charTo,
    int colname_case )
```

Open a Shapefile in write mode - extended.

Parameters

<i>shp</i>	pointer to the Shapefile object.
<i>path</i>	<i>abstract</i> pathname to the corresponding file-system files.
<i>shape</i>	the SHAPE code; expected to be one of GAIA_SHP_POINT, GAIA_SHP_POLYLINE, GAIA_SHP_POLYGON, GAIA_SHP_MULTIPPOINT, GAIA_SHP_POINTZ, GAIA_SHP_POLYLINEZ, GAIA_SHP_POLYGONZ, GAIA_SHP_MULTIPPOINTZ, GAIA_SHP_POINTM, GAIA_SHP_POLYLINEM, GAIA_SHP_POLYGONM, GAIA_SHP_MULTIPPOINTM
<i>list</i>	pointer to DBF List object representing the corresponding data attributes.
<i>charFrom</i>	GNU ICONV name identifying the input charset encoding.
<i>charTo</i>	GNU ICONV name identifying the output charset encoding.
<i>colname_case</i>	one between GAIA_DBF_COLNAME_LOWERCASE, GAIA_DBF_COLNAME_UPPERCASE or GAIA_DBF_COLNAME_CASE_IGNORE.

See also

[gaiaAllocShapefile](#), [gaiaFreeShapefile](#), [gaiaOpenShpRead](#), [gaiaReadShpEntity](#), [gaiaShpAnalyze](#), [gaiaWriteShpEntity](#), [gaiaFlushShpHeaders](#), [gaiaOpenShpWrite](#)

Note

on failure the object member *Valid* will be set to 0; and the object member *LastError* will contain the appropriate error message.

the *abstract* pathname should not contain any suffix at all.

6.10.2.57 `gaiaOpenZipDbf()`

```
GAIA GEO_DECLARE gaiaDbfPtr gaiaOpenZipDbf (
    const char * zip_path,
    const char * filename,
    const char * charFrom,
    const char * charTo )
```

Open a DBF File contained within a Zipfile (just for checking its fields)

Parameters

<i>zip_path</i>	path of the Zipfile.
<i>filename</i>	filename of the DB file within the Zipfile.
<i>charFrom</i>	GNU ICONV name identifying the input charset encoding.
<i>charTo</i>	GNU ICONV name identifying the output charset encoding.

Returns

dbf pointer to the DBF File object. NULL on error.

See also

[gaiaAllocDbf](#), [gaiaFreeDbf](#), [gaiaOpenDbfWrite](#), [gaiaReadDbfEntity](#), [gaiaWriteDbfEntity](#), [gaiaFlushDbfHeader](#)

Note

you are responsible to destroy (before or after) any allocated DBF File.
you should phisically open the DBF File in *read* or *write* mode before performing any actual I/O operation.

6.10.2.58 gaiaOutBareKml()

```
GAIA GEO_DECLARE void gaiaOutBareKml (
    gaiaOutBufferPtr out_buf,
    gaiaGeomCollPtr geom,
    int precision )
```

Encodes a Geometry object into KML notation.

Parameters

<i>out_buf</i>	pointer to dynamically growing Text buffer
<i>geom</i>	pointer to Geometry object
<i>precision</i>	decimal digits to be used for coordinates

See also

[gaiaParseKml](#), [gaiaOutFullKml](#)

Note

this function will export the simplest KML notation (no descriptions).

6.10.2.59 `gaiaOutBufferInitialize()`

```
GAIAGEO_DECLARE void gaiaOutBufferInitialize (
    gaiaOutBufferPtr buf )
```

Initializes a dynamically growing Text output buffer.

Parameters

<i>buf</i>	pointer to gaiaOutBufferStruct structure
------------	--

See also

[gaiaOutBufferReset](#), [gaiaAppendToOutBuffer](#)

Note

Text notations representing Geometry objects may easily require a huge storage amount: the [gaiaOutBufferStruct](#) automatically supports a dynamically growing output buffer. You are required to initialize this structure before attempting any further operation; and you are responsible to cleanup any related memory allocation when it's any longer required.

Examples

[demo2.c](#).

6.10.2.60 `gaiaOutBufferReset()`

```
GAIAGEO_DECLARE void gaiaOutBufferReset (
    gaiaOutBufferPtr buf )
```

Resets a dynamically growing Text output buffer to its initial (empty) state.

Parameters

<i>buf</i>	pointer to gaiaOutBufferStruct structure
------------	--

See also

[gaiaOutBufferInitialize](#), [gaiaAppendToOutBuffer](#)

Note

You are required to initialize this structure before attempting any further operation: this function will release any related memory allocation.

Examples

[demo2.c](#).

6.10.2.61 `gaiaOutFullKml()`

```
GAIAGEO_DECLARE void gaiaOutFullKml (
    gaiaOutBufferPtr out_buf,
    const char * name,
    const char * desc,
    gaiaGeomCollPtr geom,
    int precision )
```

Encodes a Geometry object into KML notation.

Parameters

<i>out_buf</i>	pointer to dynamically growing Text buffer
<i>name</i>	text string to be set as KML <i>name</i>
<i>desc</i>	text string to be set as KML <i>description</i>
<i>geom</i>	pointer to Geometry object
<i>precision</i>	decimal digits to be used for coordinates

See also

[gaiaParseKml](#), [gaiaOutBareKml](#)

Note

this function will export the simplest KML notation (no descriptions).

6.10.2.62 `gaiaOutGeoJSON()`

```
GAIAGEO_DECLARE void gaiaOutGeoJSON (
    gaiaOutBufferPtr out_buf,
    gaiaGeomCollPtr geom,
    int precision,
    int options )
```

Encodes a Geometry object into GeoJSON notation.

Parameters

<i>out_buf</i>	pointer to dynamically growing Text buffer
<i>geom</i>	pointer to Geometry object
<i>precision</i>	decimal digits to be used for coordinates
<i>options</i>	GeoJSON specific options

See also

[gaiaParseGeoJSON](#)

Note

options can assume the following values:

- 1 = BBOX, no CRS
- 2 = no BBOX, short form CRS
- 3 = BBOX, short form CRS
- 4 = no BBOX, long form CRS
- 5 = BBOX, long form CRS
- any other value: no BBOX and no CRS

6.10.2.63 gaiaOutGml()

```
GAIA GEO_DECLARE void gaiaOutGml (
    gaiaOutBufferPtr out_buf,
    int version,
    int precision,
    gaiaGeomCollPtr geom )
```

Encodes a Geometry object into GML notation.

Parameters

<i>out_buf</i>	pointer to dynamically growing Text buffer
<i>version</i>	GML version
<i>precision</i>	decimal digits to be used for coordinates
<i>geom</i>	pointer to Geometry object

See also

[gaiaParseGml](#), [gaiaOutGml_ex](#)

Note

it's simply an alias for [gaiaOutGml_ex](#) with `FLIPPED` set to `FALSE`.

6.10.2.64 gaiaOutGml_ex()

```
GAIA GEO_DECLARE void gaiaOutGml_ex (
    gaiaOutBufferPtr out_buf,
    int version,
    int flipped,
    int precision,
    gaiaGeomCollPtr geom )
```

Encodes a Geometry object into GML notation.

Parameters

<i>out_buf</i>	pointer to dynamically growing Text buffer
<i>version</i>	GML version \params flipped if TRUE the X and Y axes order will be: Y first, X second
<i>precision</i>	decimal digits to be used for coordinates
<i>geom</i>	pointer to Geometry object

See also

[gaiaParseGml](#), [gaiaOutGml](#)

Note

if *version* is set to **3**, then GMLv3 will be used; in any other case GMLv2 will be assumed by default.

6.10.2.65 **gaiaOutLinestringZ()**

```
GAIA GEO_DECLARE void gaiaOutLinestringZ (
    gaiaOutBufferPtr out_buf,
    gaiaLinestringPtr linestring )
```

Encodes a WKT 3D Linestring [XYZ].

Parameters

<i>out_buf</i>	pointer to dynamically growing Text buffer
<i>linestring</i>	pointer to Linestring object

See also

[gaiaOutPointZ](#), [gaiaOutPolygonZ](#), [gaiaOutLinestringZex](#)

Remarks

mainly intended for internal usage.

6.10.2.66 **gaiaOutLinestringZex()**

```
GAIA GEO_DECLARE void gaiaOutLinestringZex (
    gaiaOutBufferPtr out_buf,
    gaiaLinestringPtr linestring,
    int precision )
```

Encodes a WKT 3D Linestring [XYZ].

Parameters

<i>out_buf</i>	pointer to dynamically growing Text buffer
<i>linestring</i>	pointer to Linestring object
<i>precision</i>	decimal digits to be used for coordinates

See also

[gaiaOutPointZ](#), [gaiaOutPolygonZ](#)

Remarks

mainly intended for internal usage.

6.10.2.67 gaiaOutPointZ()

```
GAIAGEO_DECLARE void gaiaOutPointZ (
    gaiaOutBufferPtr out_buf,
    gaiaPointPtr point )
```

Encodes a WKT 3D Point [XYZ].

Parameters

<i>out_buf</i>	pointer to dynamically growing Text buffer
<i>point</i>	pointer to Point object

See also

[gaiaOutLinestringZ](#), [gaiaOutPolygonZ](#), [gaiaOutPointZex](#)

Remarks

mainly intended for internal usage.

6.10.2.68 gaiaOutPointZex()

```
GAIAGEO_DECLARE void gaiaOutPointZex (
    gaiaOutBufferPtr out_buf,
    gaiaPointPtr point,
    int precision )
```

Encodes a WKT 3D Point [XYZ].

Parameters

<i>out_buf</i>	pointer to dynamically growing Text buffer
<i>point</i>	pointer to Point object
<i>precision</i>	decimal digits to be used for coordinates

See also

[gaiaOutLinestringZ](#), [gaiaOutPolygonZ](#)

Remarks

mainly intended for internal usage.

6.10.2.69 gaiaOutPolygonZ()

```
GAIA GEO_DECLARE void gaiaOutPolygonZ (
    gaiaOutBufferPtr out_buf,
    gaiaPolygonPtr polygon )
```

Encodes a WKT 3D Polygon [XYZ].

Parameters

<i>out_buf</i>	pointer to dynamically growing Text buffer
<i>polygon</i>	pointer to Point object

See also

[gaiaOutPointZ](#), [gaiaOutLinestringZ](#), [gaiaOutPolygonZex](#)

Remarks

mainly intended for internal usage.

6.10.2.70 gaiaOutPolygonZex()

```
GAIA GEO_DECLARE void gaiaOutPolygonZex (
    gaiaOutBufferPtr out_buf,
    gaiaPolygonPtr polygon,
    int precision )
```

Encodes a WKT 3D Polygon [XYZ].

Parameters

<i>out_buf</i>	pointer to dynamically growing Text buffer
<i>polygon</i>	pointer to Point object
<i>precision</i>	decimal digits to be used for coordinates

See also

[gaiaOutPointZ](#), [gaiaOutLinestringZ](#)

Remarks

mainly intended for internal usage.

6.10.2.71 gaiaOutSvg()

```
GAIAGEO_DECLARE void gaiaOutSvg (
    gaiaOutBufferPtr out_buf,
    gaiaGeomCollPtr geom,
    int relative,
    int precision )
```

Encodes a Geometry object into SVG notation.

Parameters

<i>out_buf</i>	pointer to dynamically growing Text buffer
<i>geom</i>	pointer to Geometry object
<i>relative</i>	flag: relative or absolute coordinates
<i>precision</i>	decimal digits to be used for coordinates

Note

if *relative* is set to 1, then SVG relative coords will be used: in any other case SVG absolute coords will be assumed by default.

6.10.2.72 gaiaOutWkt()

```
GAIAGEO_DECLARE void gaiaOutWkt (
    gaiaOutBufferPtr out_buf,
    gaiaGeomCollPtr geom )
```

Encodes a Geometry object into WKT notation.

Parameters

<i>out_buf</i>	pointer to dynamically growing Text buffer
<i>geom</i>	pointer to Geometry object

See also

[gaiaParseWkt](#), [gaiaOutWktStrict](#), [gaiaParseEWKT](#), [gaiaToEWKT](#), [gaiaOutWktEx](#)

Note

this function will apply 3D WKT encoding as internally intended by Spatialite: not necessarily intended by other OGC-like implementations.
Anyway, 2D WKT is surely standard and safely interoperable.

Examples

[demo2.c](#).

6.10.2.73 gaiaOutWktEx()

```
GAIA GEO_DECLARE void gaiaOutWktEx (
    gaiaOutBufferPtr out_buf,
    gaiaGeomCollPtr geom,
    int precision )
```

Encodes a Geometry object into WKT notation.

Parameters

<i>out_buf</i>	pointer to dynamically growing Text buffer
<i>geom</i>	pointer to Geometry object
<i>precision</i>	decimal digits to be used for coordinates

See also

[gaiaParseWkt](#), [gaiaOutWktStrict](#), [gaiaParseEWKT](#), [gaiaToEWKT](#)

Note

this function will apply 3D WKT encoding as internally intended by Spatialite: not necessarily intended by other OGC-like implementations.
Anyway, 2D WKT is surely standard and safely interoperable.

6.10.2.74 `gaiaOutWktStrict()`

```
GAIA GEO_DECLARE void gaiaOutWktStrict (
    gaiaOutBufferPtr out_buf,
    gaiaGeomCollPtr geom,
    int precision )
```

Encodes a Geometry object into strict 2D WKT notation.

Parameters

<i>out_buf</i>	pointer to dynamically growing Text buffer
<i>geom</i>	pointer to Geometry object
<i>precision</i>	decimal digits to be used for coordinates

See also

[gaiaParseWkt](#), [gaiaOutWkt](#), [gaiaParseEWKT](#), [gaiaToEWKT](#)

Note

this function will apply strict 2D WKT encoding, so to be surely standard and safely interoperable. Dimensions will be automatically casted to 2D [XY] when required.

6.10.2.75 `gaiaParseEWKT()`

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaParseEWKT (
    const unsigned char * in_buffer )
```

Creates a Geometry object from EWKT notation.

Parameters

<i>in_buffer</i>	pointer to EWKT buffer
------------------	------------------------

Returns

the pointer to the newly created Geometry object: NULL on failure

See also

[gaiaParseWkt](#), [gaiaOutWkt](#), [gaiaOutWktStrict](#), [gaiaToEWKT](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, unless you've passed ownership of the Geometry object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.10.2.76 `gaiaParseGeoJSON()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaParseGeoJSON (
    const unsigned char * in_buffer )
```

Creates a Geometry object from GeoJSON notation.

Parameters

<i>in_buffer</i>	pointer to GeoJSON buffer
------------------	---------------------------

Returns

the pointer to the newly created Geometry object: NULL on failure

See also

[gaiaOutGeoJSON](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, unless you've passed ownership of the Geometry object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.10.2.77 `gaiaParseGml()`

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaParseGml (
    const unsigned char * in_buffer,
    sqlite3 * sqlite_handle )
```

Creates a Geometry object from GML notation.

Parameters

<i>in_buffer</i>	pointer to GML buffer
<i>sqlite_handle</i>	handle to current DB connection

Returns

the pointer to the newly created Geometry object: NULL on failure

See also

[gaiaParseGml_r](#), [gaiaOutGml](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, unless you've passed ownership of the Geometry object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.
not reentrant and thread unsafe.

6.10.2.78 gaiaParseGml_r()

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaParseGml_r (
    const void * p_cache,
    const unsigned char * in_buffer,
    sqlite3 * sqlite_handle )
```

Creates a Geometry object from GML notation.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>in_buffer</i>	pointer to GML buffer
<i>sqlite_handle</i>	handle to current DB connection

Returns

the pointer to the newly created Geometry object: NULL on failure

See also

[gaiaParseGml](#), [gaiaOutGml](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, unless you've passed ownership of the Geometry object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.
reentrant and thread-safe.

6.10.2.79 gaiaParseHexEWKB()

```
GAIA GEO_DECLARE unsigned char* gaiaParseHexEWKB (
    const unsigned char * blob_hex,
    int * blob_size )
```

Translates an EWKB notation from hexadecimal into binary.

Parameters

<i>blob_hex</i>	pointer to EWKB input buffer (hexadecimal text string)
<i>blob_size</i>	length (in bytes) of the input buffer; if succesfull will contain the length of the returned output buffer.

Returns

the pointer to the newly created EWKB binary buffer: NULL on failure.

See also

[gaiaToWkb](#), [gaiaToHexWkb](#), [gaiaFromEWKB](#), [gaiaToEWKB](#)

Note

you are responsible to destroy (before or after) any buffer allocated by [gaiaParseHexEWKB\(\)](#)

6.10.2.80 gaiaParseKml()

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaParseKml (  
    const unsigned char * in_buffer )
```

Creates a Geometry object from KML notation.

Parameters

<i>in_buffer</i>	pointer to KML buffer
------------------	-----------------------

Returns

the pointer to the newly created Geometry object: NULL on failure

See also

[gaiaOutBareKml](#), [gaiaOutFullKml](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, unless you've passed ownership of the Geometry object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.10.2.81 `gaiaParseWkt()`

```
GAIA GEO_DECLARE gaiaGeomCollPtr gaiaParseWkt (
    const unsigned char * in_buffer,
    short type )
```

Creates a Geometry object from WKT notation.

Parameters

<i>in_buffer</i>	pointer to WKT buffer
<i>type</i>	the expected Geometry Class Type if actual type defined in WKT doesn't corresponds to this, an error will be raised.

Returns

the pointer to the newly created Geometry object: NULL on failure

See also

[gaiaOutWkt](#), [gaiaOutWktStrict](#), [gaiaParseEWKT](#), [gaiaToEWKT](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, unless you've passed ownership of the Geometry object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.10.2.82 `gaiaReadDbfEntity()`

```
GAIA GEO_DECLARE int gaiaReadDbfEntity (
    gaiaDbfPtr dbf,
    int current_row,
    int * deleted )
```

Reads a record from a DBF File object.

Parameters

<i>dbf</i>	pointer to the DBF File object.
<i>current_row</i>	the row number identifying the record to be read.
<i>deleted</i>	on completion this variable will contain 0 if the record just read is valid: any other value if the record just read is marked as <i>logically deleted</i> .

Returns

0 on failure: any other value on success.

See also

[gaiaAllocDbf](#), [gaiaFreeDbf](#), [gaiaOpenDbfRead](#), [gaiaOpenDbfWrite](#), [gaiaFlushDbfHeader](#)

Note

on completion the DBF File *First* member will point to the linked list containing the corresponding data attributes [both data formats and values].

Remarks

the DBF File object should be opened in *read* mode.

6.10.2.83 gaiaReadDbfEntity_ex()

```
GAIGEO_DECLARE int gaiaReadDbfEntity_ex (
    gaiaDbfPtr dbf,
    int current_row,
    int * deleted,
    int text_dates )
```

Reads a record from a DBF File object.

Parameters

<i>dbf</i>	pointer to the DBF File object.
<i>current_row</i>	the row number identifying the record to be read.
<i>deleted</i>	on completion this variable will contain 0 if the record
<i>text_dates</i>	is TRUE all DBF dates will be considered as TEXT just read is valid: any other value if the record just read is marked as <i>logically deleted</i> .

Returns

0 on failure: any other value on success.

See also

[gaiaAllocDbf](#), [gaiaFreeDbf](#), [gaiaOpenDbfRead](#), [gaiaOpenDbfWrite](#), [gaiaFlushDbfHeader](#)

Note

on completion the DBF File *First* member will point to the linked list containing the corresponding data attributes [both data formats and values].

Remarks

the DBF File object should be opened in *read* mode.

6.10.2.84 gaiaReadShpEntity()

```
GAIGEO_DECLARE int gaiaReadShpEntity (
    gaiaShapefilePtr shp,
    int current_row,
    int srid )
```

Reads a feature from a Shapefile object.

Parameters

<i>shp</i>	pointer to the Shapefile object.
<i>current_row</i>	the row number identifying the feature to be read.
<i>srid</i>	feature's SRID

Returns

0 on failure: any other value on success. -1 when the corresponding DBF record is marked as DELETED.

See also

[gaiaAllocShapefile](#), [gaiaFreeShapefile](#), [gaiaOpenShpRead](#), [gaiaOpenShpWrite](#), [gaiaShpAnalyze](#), [gaiaWriteShpEntity](#), [gaiaFlushShpHeaders](#)

Note

on completion the Shapefile's *Dbf* member will contain the feature read:

- the *Dbf->Geometry* member will contain the corresponding Geometry
- and the *Dbf->First* member will point to the linked list containing the corresponding data attributes [both data formats and values].

Remarks

the Shapefile object should be opened in *read* mode.

6.10.2.85 **gaiaReadShpEntity_ex()**

```
GAIA GEO_DECLARE int gaiaReadShpEntity_ex (
    gaiaShapefilePtr shp,
    int current_row,
    int srid,
    int text_dates )
```

Reads a feature from a Shapefile object.

Parameters

<i>shp</i>	pointer to the Shapefile object.
<i>current_row</i>	the row number identifying the feature to be read.
<i>srid</i>	feature's SRID
<i>text_dates</i>	is TRUE all DBF dates will be considered as TEXT

Returns

0 on failure: any other value on success.

See also

[gaiaAllocShapefile](#), [gaiaFreeShapefile](#), [gaiaOpenShpRead](#), [gaiaOpenShpWrite](#), [gaiaShpAnalyze](#), [gaiaWriteShpEntity](#), [gaiaFlushShpHeaders](#)

Note

on completion the Shapefile's *Dbf* member will contain the feature read:

- the *Dbf->Geometry* member will contain the corresponding Geometry
- and the *Dbf->First* member will point to the linked list containing the corresponding data attributes [both data formats and values].

Remarks

the Shapefile object should be opened in *read* mode.

6.10.2.86 gaiaReadWktFromZipShp()

```
GAIA GEO_DECLARE char* gaiaReadWktFromZipShp (
    const char * zip_path,
    const char * basename )
```

Attempting to get a WKT from the .PRJ member of a given zipped Shapefile.

Parameters

<i>zip_path</i>	absolute or relativ pathname leading to the Zipfile.
<i>basename</i>	name of the Shapefile (excluding any .shp, .shx or .dbf suffix)

Returns

the returned WKT expression from the .prj member of the zipped Shapefile or NULL on failure

Note

the WKT expression corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

6.10.2.87 gaiaResetDbfEntity()

```
GAIA GEO_DECLARE void gaiaResetDbfEntity (
    gaiaDbfListPtr list )
```

Resets a DBF List object to its initial empty state.

Parameters

<i>list</i>	pointer to the DBF List object.
-------------	---------------------------------

See also

[gaiaFreeValue](#)

Note

any DBF Field associated to the List object will be reset to its initial empty state (i.e. *no value* at all).

6.10.2.88 gaiaSetDoubleValue()

```
GAIAGEO_DECLARE void gaiaSetDoubleValue (
    gaiaDbfFieldPtr field,
    double value )
```

Sets a DOUBLE current value for a DBF Field object.

Parameters

<i>field</i>	pointer to DBF Field object.
<i>value</i>	double value to be set.

See also

[gaiaAllocDbfField](#), [gaiaFreeDbfField](#), [gaiaCloneDbfField](#), [gaiaFreeValue](#), [gaiaSetNullValue](#), [gaiaSetIntValue](#), [gaiaSetStrValue](#)

6.10.2.89 gaiaSetIntValue()

```
GAIAGEO_DECLARE void gaiaSetIntValue (
    gaiaDbfFieldPtr field,
    sqlite3_int64 value )
```

Sets an INTEGER current value for a DBF Field object.

Parameters

<i>field</i>	pointer to DBF Field object.
<i>value</i>	integer value to be set.

See also

[gaiaAllocDbfField](#), [gaiaFreeDbfField](#), [gaiaCloneDbfField](#), [gaiaFreeValue](#), [gaiaSetNullValue](#), [gaiaSetDoubleValue](#), [gaiaSetStrValue](#)

6.10.2.90 **gaiaSetNullValue()**

```
GAIAGEO_DECLARE void gaiaSetNullValue (
    gaiaDbfFieldPtr field )
```

Sets a NULL current value for a DBF Field object.

Parameters

<i>field</i>	pointer to DBF Field object
--------------	-----------------------------

See also

[gaiaAllocDbfField](#), [gaiaFreeDbfField](#), [gaiaCloneDbfField](#), [gaiaFreeValue](#), [gaiaSetIntValue](#), [gaiaSetDoubleValue](#), [gaiaSetStrValue](#)

6.10.2.91 **gaiaSetStrValue()**

```
GAIAGEO_DECLARE void gaiaSetStrValue (
    gaiaDbfFieldPtr field,
    char * str )
```

Sets a TEXT current value for a DBF Field object.

Parameters

<i>field</i>	pointer to DBF Field object.
<i>str</i>	text string value to be set.

See also

[gaiaAllocDbfField](#), [gaiaFreeDbfField](#), [gaiaCloneDbfField](#), [gaiaFreeValue](#), [gaiaSetNullValue](#), [gaiaSetIntValue](#), [gaiaSetDoubleValue](#)

6.10.2.92 **gaiaShpAnalyze()**

```
GAIAGEO_DECLARE void gaiaShpAnalyze (
    gaiaShapefilePtr shp )
```

Prescans a Shapefile object gathering informations.

Parameters

<i>shp</i>	pointer to the Shapefile object.
------------	----------------------------------

See also

[gaiaAllocShapefile](#), [gaiaFreeShapefile](#), [gaiaOpenShpRead](#), [gaiaOpenShpWrite](#), [gaiaReadShpEntity](#), [gaiaWriteShpEntity](#), [gaiaFlushShpHeaders](#)

Note

on completion the Shapefile's *EffectiveType* will contain the Geometry type corresponding to features actually found.

Remarks

the Shapefile object should be opened in *read* mode.

6.10.2.93 **gaiaTextReaderAlloc()**

```
GAIA GEO_DECLARE gaiaTextReaderPtr gaiaTextReaderAlloc (
    const char * path,
    char field_separator,
    char text_separator,
    char decimal_separator,
    int first_line_titles,
    const char * encoding )
```

Creates a Text Reader object.

Parameters

<i>path</i>	to the corresponding file-system file.
<i>field_separator</i>	the character acting as a separator between adjacent fields.
<i>text_separator</i>	the character used to quote text strings.
<i>decimal_separator</i>	the character used as a separator between integer and decimal digits for real numeric values.
<i>first_line_titles</i>	0 if the first line contains regular values: any other value if the first line contains column names.
<i>encoding</i>	GNU ICONV name identifying the input charset encoding.

Returns

the pointer to the newly created Text Reader object: NULL on failure

See also

[gaiaTextReaderDestroy](#), [gaiaTextReaderParse](#), [gaiaTextReaderGetRow](#), [gaiaTextReaderFetchField](#)

Note

you are responsible to destroy (before or after) any allocated Text Reader object.

6.10.2.94 gaiaTextReaderDestroy()

```
GAIA GEO_DECLARE void gaiaTextReaderDestroy (
    gaiaTextReaderPtr reader )
```

Destroys a Text Reader object.

Parameters

<i>reader</i>	pointer to Text Reader object.
---------------	--------------------------------

See also

[gaiaTextReaderAlloc](#), [gaiaTextReaderParse](#), [gaiaTextReaderGetRow](#), [gaiaTextReaderFetchField](#)

6.10.2.95 gaiaTextReaderFetchField()

```
GAIA GEO_DECLARE int gaiaTextReaderFetchField (
    gaiaTextReaderPtr reader,
    int field_num,
    int * type,
    const char ** value )
```

Retrieves an individual field value from the current Line.

Parameters

<i>reader</i>	pointer to Text Reader object.
<i>field_num</i>	relative field [aka column] index: first field has index 0.
<i>type</i>	on completion this variable will contain the value type.
<i>value</i>	on completion this variable will contain the current field value.

Returns

0 on failure: any other value on success.

See also

[gaiaTextReaderAlloc](#), [gaiaTextReaderDestroy](#), [gaiaTextReaderParse](#), [gaiaTextReaderGetRow](#)

6.10.2.96 `gaiaTextReaderGetRow()`

```
GAIA GEO_DECLARE int gaiaTextReaderGetRow (
    gaiaTextReaderPtr reader,
    int row_num )
```

Reads a line from a Text Reader object.

Parameters

<i>reader</i>	pointer to Text Reader object.
<i>row_num</i>	the Line Number identifying the Line to be read.

Returns

0 on failure: any other value on success.

See also

[gaiaTextReaderAlloc](#), [gaiaTextReaderDestroy](#), [gaiaTextReaderParse](#), [gaiaTextReaderFetchField](#)

Note

this function will load the requested Line into the current buffer: you can then use [gaiaTextReaderFetchField](#) in order to retrieve any individual field [aka column] value.

6.10.2.97 `gaiaTextReaderParse()`

```
GAIA GEO_DECLARE int gaiaTextReaderParse (
    gaiaTextReaderPtr reader )
```

Prescans the external file associated to a Text Reader object.

Parameters

<i>reader</i>	pointer to Text Reader object.
---------------	--------------------------------

Returns

0 on failure: any other value on success.

See also

[gaiaTextReaderAlloc](#), [gaiaTextReaderDestroy](#), [gaiaTextReaderGetRow](#), [gaiaTextReaderFetchField](#)

Note

this preliminary step is required so to ensure:

- file consistency: checking expected formatting rules.
- identifying the number / type / name of fields [aka columns].
- identifying the actual number of lines within the file.

6.10.2.98 gaiaToCompressedBlobWkb()

```
GAIA GEO_DECLARE void gaiaToCompressedBlobWkb (
    gaiaGeomCollPtr geom,
    unsigned char ** result,
    int * size )
```

Creates a Compressed BLOB-Geometry corresponding to a Geometry object.

Parameters

<i>geom</i>	pointer to the Geometry object.
<i>result</i>	on completion will contain a pointer to Compressed BLOB-Geometry: NULL on failure.
<i>size</i>	on completion this variable will contain the BLOB's size (in bytes)

See also

[gaiaFromSpatialiteBlobWkb](#), [gaiaToSpatialiteBlobWkb](#)

Note

this function will apply compression to any Linestring / Ring found within the Geometry to be encoded.
the returned BLOB buffer corresponds to dynamically allocated memory: so you are responsible to free() it
[unless SQLite will take care of memory cleanup via buffer binding].

6.10.2.99 gaiaToEWKB()

```
GAIA GEO_DECLARE void gaiaToEWKB (
    gaiaOutBufferPtr out_buf,
    gaiaGeomCollPtr geom )
```

Encodes a Geometry object into EWKB notation.

Parameters

<i>out_buf</i>	pointer to dynamically growing Text buffer
<i>geom</i>	pointer to Geometry object

See also

[gaiaFromWkb](#), [gaiaToWkb](#), [gaiaToHexWkb](#), [gaiaFromEWKB](#), [gaiaToEWKB](#)

Note

this function will produce strictly conformant EWKB; you can safely use this for PostGIS data exchange.

6.10.2.100 gaiaToEWKT()

```
GAIAGEO_DECLARE void gaiaToEWKT (
    gaiaOutBufferPtr out_buf,
    gaiaGeomCollPtr geom )
```

Encodes a Geometry object into EWKT notation.

Parameters

<i>out_buf</i>	pointer to dynamically growing Text buffer
<i>geom</i>	pointer to Geometry object

See also

[gaiaParseWkt](#), [gaiaOutWkt](#), [gaiaOutWktStrict](#), [gaiaParseEWKT](#)

Note

this function will apply PostGIS own EWKT encoding.

6.10.2.101 gaiaToFgf()

```
GAIAGEO_DECLARE void gaiaToFgf (
    gaiaGeomCollPtr geom,
    unsigned char ** result,
    int * size,
    int coord_dims )
```

Encodes a Geometry object into FGF notation.

Parameters

<i>geom</i>	pointer to Geometry object
<i>result</i>	on completion will contain a pointer to the FGF buffer [BLOB]; NULL on failure.
<i>size</i>	on completion this variable will contain the BLOB's size (in bytes)
<i>coord_dims</i>	one of: GAIA_XY, GAIA_XY_Z, GAIA_XY_M, GAIA_XY_ZM

See also

[gaiaFromFgf](#)

Note

the returned BLOB buffer corresponds to dynamically allocated memory: so you are responsible to free() it [unless SQLite will take care of memory cleanup via buffer binding].

6.10.2.102 `gaiaToHexWkb()`

```
GAIA_GEO_DECLARE char* gaiaToHexWkb (
    gaiaGeomCollPtr geom )
```

Encodes a Geometry object into (hex) WKB notation.

Parameters

<i>geom</i>	pointer to Geometry object
-------------	----------------------------

Returns

the pointer to a text buffer containing WKB translated into plain hexadecimal: NULL on failure.

See also

[gaiaFromWkb](#), [gaiaToWkb](#), [gaiaFromEWKB](#), [gaiaToEWKB](#)

Note

the returned buffer corresponds to dynamically allocated memory: so you are responsible to free() it [unless SQLite will take care of memory cleanup via buffer binding].

6.10.2.103 `gaiaToSpatialiteBlobWkb()`

```
GAIA_GEO_DECLARE void gaiaToSpatialiteBlobWkb (
    gaiaGeomCollPtr geom,
    unsigned char ** result,
    int * size )
```

Creates a BLOB-Geometry corresponding to a Geometry object.

Parameters

<i>geom</i>	pointer to the Geometry object.
<i>result</i>	on completion will contain a pointer to BLOB-Geometry: NULL on failure.
<i>size</i>	on completion this variable will contain the BLOB's size (in bytes)

See also

[gaiaFromSpatialiteBlobWkb](#), [gaiaToCompressedBlobWkb](#)

Note

the BLOB buffer corresponds to dynamically allocated memory: so you are responsible to free() it [unless SQLite will take care of memory cleanup via buffer binding].

Examples

[demo3.c](#), and [demo4.c](#).

6.10.2.104 `gaiaToSpatialiteBlobWkbEx()`

```
GAIA GEO_DECLARE void gaiaToSpatialiteBlobWkbEx (
    gaiaGeomCollPtr geom,
    unsigned char ** result,
    int * size,
    int gpkg_mode )
```

Creates a BLOB-Geometry corresponding to a Geometry object.

Parameters

<i>geom</i>	pointer to the Geometry object.
<i>result</i>	on completion will contain a pointer to BLOB-Geometry: NULL on failure.
<i>size</i>	on completion this variable will contain the BLOB's size (in bytes)
<i>gpkg_mode</i>	if set to TRUE will always return GPKG Geometry-BLOBs

See also

[gaiaFromSpatialiteBlobWkb](#), [gaiaToCompressedBlobWkb](#), [gaiaToSpatialiteBlobWkbEx2](#)

Note

the BLOB buffer corresponds to dynamically allocated memory: so you are responsible to free() it [unless SQLite will take care of memory cleanup via buffer binding].

6.10.2.105 `gaiaToSpatialiteBlobWkbEx2()`

```
GAIA GEO_DECLARE void gaiaToSpatialiteBlobWkbEx2 (
    gaiaGeomCollPtr geom,
    unsigned char ** result,
    int * size,
    int gpkg_mode,
    int tiny_point )
```

Creates a BLOB-Geometry corresponding to a Geometry object.

Parameters

<i>geom</i>	pointer to the Geometry object.
<i>result</i>	on completion will contain a pointer to BLOB-Geometry: NULL on failure.
<i>size</i>	on completion this variable will contain the BLOB's size (in bytes)
<i>gpkg_mode</i>	if set to TRUE will always return GPKG Geometry-BLOBs
<i>tiny_point</i>	if set to TRUE all POINT Geometries will be encoded by using the TinyPoint BLOB format.

See also

[gaiaFromSpatialiteBlobWkb](#), [gaiaToCompressedBlobWkb](#), [gaiaToSpatialiteBlobWkbEx](#)

Note

the BLOB buffer corresponds to dynamically allocated memory: so you are responsible to free() it [unless SQLite will take care of memory cleanup via buffer binding].

6.10.2.106 **gaiaToWkb()**

```
GAIA_GEO_DECLARE void gaiaToWkb (
    gaiaGeomCollPtr geom,
    unsigned char ** result,
    int * size )
```

Encodes a Geometry object into WKB notation.

Parameters

<i>geom</i>	pointer to Geometry object
<i>result</i>	on completion will contain a pointer to the WKB buffer [BLOB]: NULL on failure.
<i>size</i>	on completion this variable will contain the BLOB's size (in bytes)

See also

[gaiaFromWkb](#), [gaiaToHexWkb](#), [gaiaFromEWKB](#), [gaiaToEWKB](#)

Note

this function will apply 3D WKB encoding as internally intended by Spatialite: not necessarily intended by other OGC-like implementations.

Anyway, 2D WKB is surely standard and safely interoperable.

the returned BLOB buffer corresponds to dynamically allocated memory: so you are responsible to free() it [unless SQLite will take care of memory cleanup via buffer binding].

6.10.2.107 `gaiaWriteDbfEntity()`

```
GAIA GEO_DECLARE int gaiaWriteDbfEntity (
    gaiaDbfPtr dbf,
    gaiaDbfListPtr entity )
```

Writes a record into a DBF File object.

Parameters

<i>dbf</i>	pointer to the DBF File object.
<i>entity</i>	pointer to DBF List object containing Fields and corresponding values.

Returns

0 on failure: any other value on success.

See also

[gaiaAllocDbf](#), [gaiaFreeDbf](#), [gaiaOpenDbfRead](#), [gaiaOpenDbfWrite](#), [gaiaReadDbfEntity](#), [gaiaFlushDbfHeader](#)

Remarks

the DBF File object should be opened in *write* mode.

6.10.2.108 `gaiaWriteShpEntity()`

```
GAIA GEO_DECLARE int gaiaWriteShpEntity (
    gaiaShapefilePtr shp,
    gaiaDbfListPtr entity )
```

Writes a feature into a Shapefile object.

Parameters

<i>shp</i>	pointer to the Shapefile object.
<i>entity</i>	pointer to DBF List object containing both Geometry and Field values.

Returns

0 on failure: any other value on success.

See also

[gaiaAllocShapefile](#), [gaiaFreeShapefile](#), [gaiaOpenShpRead](#), [gaiaOpenShpWrite](#), [gaiaReadShpEntity](#), [gaiaShpAnalyze](#), [gaiaFlushShpHeaders](#)

Remarks

the Shapefile object should be opened in *write* mode.

6.10.2.109 gaiaZipfileDbfN()

```
GAIA GEO_DECLARE char* gaiaZipfileDbfN (
    const char * zip_path,
    int idx )
```

Will return the filename of the Nth DBF file from within a given Zipfile.

Parameters

<i>zip_path</i>	absolute or relativ pathname leading to the Zipfile.
<i>idx</i>	index (1-based) of the DBF file

Returns

the filename of the Nth DBF file or NULL on failure

Note

the returned filename corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

6.10.2.110 gaiaZipfileNumDBF()

```
GAIA GEO_DECLARE int gaiaZipfileNumDBF (
    const char * zip_path,
    int * count )
```

Will return the number of DBF files from within a given Zipfile.

Parameters

<i>zip_path</i>	absolute or relativ pathname leading to the Zipfile.
<i>count</i>	on success will contain the number of DBF files

Returns

0 on failure; any other value on success.

6.10.2.111 gaiaZipfileNumSHP()

```
GAIA GEO_DECLARE int gaiaZipfileNumSHP (
    const char * zip_path,
    int * count )
```

Will return the number of Shapefiles from within a given Zipfile.

Parameters

<i>zip_path</i>	absolute or relative pathname leading to the Zipfile.
<i>count</i>	on success will contain the number of Shapefiles

Returns

0 on failure; any other value on success.

6.10.2.112 gaiaZipfileShpN()

```
GAIA GEO_DECLARE char* gaiaZipfileShpN (
    const char * zip_path,
    int idx )
```

Will return the basename of the Nth Shapefile from within a given Zipfile.

Parameters

<i>zip_path</i>	absolute or relative pathname leading to the Zipfile.
<i>idx</i>	index (1-based) of the Shapefile

Returns

the basename of the Nth Shapefile or NULL on failure

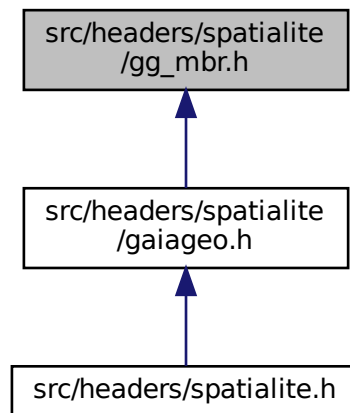
Note

the returned basename corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

6.11 src/headers/spatialite/gg_mbr.h File Reference

Geometry handling functions: MBR.

This graph shows which files directly or indirectly include this file:



Functions

- GAIAGEO_DECLARE void [gaiaMbrLinestring](#) ([gaiaLinestringPtr](#) line)
Updates the actual MBR for a Linestring object.
- GAIAGEO_DECLARE void [gaiaMbrRing](#) ([gaiaRingPtr](#) rng)
Updates the actual MBR for a Ring object.
- GAIAGEO_DECLARE void [gaiaMbrPolygon](#) ([gaiaPolygonPtr](#) polyg)
Updates the actual MBR for a Polygon object.
- GAIAGEO_DECLARE void [gaiaMbrGeometry](#) ([gaiaGeomCollPtr](#) geom)
Updates the actual MBR for a Geometry object.
- GAIAGEO_DECLARE int [gaiaGetMbrMinX](#) (const unsigned char *blob, unsigned int size, double *minx)
Retrieves the MBR (MinX) from a BLOB-Geometry object.
- GAIAGEO_DECLARE int [gaiaGetMbrMaxX](#) (const unsigned char *blob, unsigned int size, double *maxx)
Retrieves the MBR (MaxX) from a BLOB-Geometry object.
- GAIAGEO_DECLARE int [gaiaGetMbrMinY](#) (const unsigned char *blob, unsigned int size, double *miny)
Retrieves the MBR (MinY) from a BLOB-Geometry object.
- GAIAGEO_DECLARE int [gaiaGetMbrMaxY](#) (const unsigned char *blob, unsigned int size, double *maxy)
Retrieves the MBR (MaxY) from a BLOB-Geometry object.
- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaFromSpatialiteBlobMbr](#) (const unsigned char *blob, unsigned int size)
Creates a Geometry object corresponding to the Envelope [MBR] for a BLOB-Geometry.
- GAIAGEO_DECLARE int [gaiaMbrsContains](#) ([gaiaGeomCollPtr](#) mbr1, [gaiaGeomCollPtr](#) mbr2)
MBRs comparison: Contains.
- GAIAGEO_DECLARE int [gaiaMbrsDisjoint](#) ([gaiaGeomCollPtr](#) mbr1, [gaiaGeomCollPtr](#) mbr2)
MBRs comparison: Disjoint.
- GAIAGEO_DECLARE int [gaiaMbrsEqual](#) ([gaiaGeomCollPtr](#) mbr1, [gaiaGeomCollPtr](#) mbr2)
MBRs comparison: Equal.
- GAIAGEO_DECLARE int [gaiaMbrsIntersects](#) ([gaiaGeomCollPtr](#) mbr1, [gaiaGeomCollPtr](#) mbr2)
MBRs comparison: Intersects.

- GAIAGEO_DECLARE int [gaiaMbrsOverlaps](#) ([gaiaGeomCollPtr](#) mbr1, [gaiaGeomCollPtr](#) mbr2)
MBRs comparison: Overlaps.
- GAIAGEO_DECLARE int [gaiaMbrsTouches](#) ([gaiaGeomCollPtr](#) mbr1, [gaiaGeomCollPtr](#) mbr2)
MBRs comparison: Touches.
- GAIAGEO_DECLARE int [gaiaMbrsWithin](#) ([gaiaGeomCollPtr](#) mbr1, [gaiaGeomCollPtr](#) mbr2)
MBRs comparison: Within.
- GAIAGEO_DECLARE void [gaiaBuildMbr](#) (double x1, double y1, double x2, double y2, int srid, unsigned char **result, int *size)
Creates a BLOB-Geometry representing an Envelope [MBR].
- GAIAGEO_DECLARE void [gaiaBuildCircleMbr](#) (double x, double y, double radius, int srid, unsigned char **result, int *size)
Creates a BLOB-Geometry representing an Envelope [MBR].
- GAIAGEO_DECLARE void [gaiaBuildFilterMbr](#) (double x1, double y1, double x2, double y2, int mode, unsigned char **result, int *size)
Creates a BLOB-FilterMBR.
- GAIAGEO_DECLARE int [gaiaParseFilterMbr](#) (unsigned char *result, int size, double *minx, double *miny, double *maxx, double *maxy, int *mode)
Creates a BLOB-FilterMBR.
- GAIAGEO_DECLARE void [gaiaZRangeLinestring](#) ([gaiaLinestringPtr](#) line, double *min, double *max)
Computes the Z-Range for a Linestring object.
- GAIAGEO_DECLARE void [gaiaZRangeLinestringEx](#) ([gaiaLinestringPtr](#) line, double nodata, double *min, double *max)
Computes the Z-Range for a Linestring object (NODATA flavor)
- GAIAGEO_DECLARE void [gaiaZRangeRing](#) ([gaiaRingPtr](#) rng, double *min, double *max)
Computes the Z-Range for a Ring object.
- GAIAGEO_DECLARE void [gaiaZRangeRingEx](#) ([gaiaRingPtr](#) rng, double nodata, double *min, double *max)
Computes the Z-Range for a Ring object (NODATA flavor)
- GAIAGEO_DECLARE void [gaiaZRangePolygon](#) ([gaiaPolygonPtr](#) polyg, double *min, double *max)
Computes the Z-Range for a Polygon object.
- GAIAGEO_DECLARE void [gaiaZRangePolygonEx](#) ([gaiaPolygonPtr](#) polyg, double nodata, double *min, double *max)
Computes the Z-Range for a Polygon object (NODATA flavor)
- GAIAGEO_DECLARE void [gaiaZRangeGeometry](#) ([gaiaGeomCollPtr](#) geom, double *min, double *max)
Computes the Z-Range for a Geometry object.
- GAIAGEO_DECLARE void [gaiaZRangeGeometryEx](#) ([gaiaGeomCollPtr](#) geom, double nodata, double *min, double *max)
Computes the Z-Range for a Geometry object (NODATA flavor)
- GAIAGEO_DECLARE void [gaiaMRangeLinestring](#) ([gaiaLinestringPtr](#) line, double *min, double *max)
Computes the M-Range for a Linestring object.
- GAIAGEO_DECLARE void [gaiaMRangeLinestringEx](#) ([gaiaLinestringPtr](#) line, double nodata, double *min, double *max)
Computes the M-Range for a Linestring object (NODATA flavor)
- GAIAGEO_DECLARE void [gaiaMRangeRing](#) ([gaiaRingPtr](#) rng, double *min, double *max)
Computes the M-Range for a Ring object.
- GAIAGEO_DECLARE void [gaiaMRangeRingEx](#) ([gaiaRingPtr](#) rng, double nodata, double *min, double *max)
Computes the M-Range for a Ring object (NODATA flavor)
- GAIAGEO_DECLARE void [gaiaMRangePolygon](#) ([gaiaPolygonPtr](#) polyg, double *min, double *max)
Computes the M-Range for a Polygon object.
- GAIAGEO_DECLARE void [gaiaMRangePolygonEx](#) ([gaiaPolygonPtr](#) polyg, double nodata, double *min, double *max)
Computes the M-Range for a Polygon object (NODATA flavor)
- GAIAGEO_DECLARE void [gaiaMRangeGeometry](#) ([gaiaGeomCollPtr](#) geom, double *min, double *max)

Computes the Z-Range for a Geometry object.

- GAIAGEO_DECLARE void [gaiaMRangeGeometryEx](#) ([gaiaGeomCollPtr](#) geom, double nodata, double *min, double *max)

Computes the Z-Range for a Geometry object (NODATA flavor)

6.11.1 Detailed Description

Geometry handling functions: MBR.

6.11.2 Function Documentation

6.11.2.1 [gaiaBuildCircleMbr\(\)](#)

```
GAIAGEO_DECLARE void gaiaBuildCircleMbr (
    double x,
    double y,
    double radius,
    int srid,
    unsigned char ** result,
    int * size )
```

Creates a BLOB-Geometry representing an Envelope [MBR].

Parameters

<i>x</i>	centre X coordinate.
<i>y</i>	centre Y coordinate.
<i>radius</i>	the radius of the circle
<i>srid</i>	the SRID associated to the Envelope
<i>result</i>	on completion will contain a pointer to newly created BLOB-Geometry
<i>size</i>	on completion this variable will contain the BLOB's size (in bytes)

See also

[gaiaBuildMbr](#)

Note

the *circle* of given *radius* and *centre* will be used so to determine the corresponding *square* Envelope

6.11.2.2 gaiaBuildFilterMbr()

```
GAIA GEO_DECLARE void gaiaBuildFilterMbr (
    double x1,
    double y1,
    double x2,
    double y2,
    int mode,
    unsigned char ** result,
    int * size )
```

Creates a BLOB-FilterMBR.

Parameters

<i>x1</i>	first X coordinate.
<i>y1</i>	first Y coordinate.
<i>x2</i>	second X coordinate.
<i>y2</i>	second Y coordinate.
<i>mode</i>	one of: GAIA_FILTER_MBR_WITHIN, GAIA_FILTER_MBR_CONTAINS, GAIA_FILTER_MBR_INTERSECTS, GAIA_FILTER_MBR_DECLARE
<i>result</i>	on completion will contain a pointer to newly created BLOB-FilterMBR
<i>size</i>	on completion this variable will contain the BLOB's size (in bytes)

See also

[gaiaParseFilterMbr](#)

Note

[XY] coords must define two extreme Points identifying a diagonal of the MBR [Envelope]
no special order is required for coords: MAX / MIN values will be internally arranged as appropriate.

Remarks

internally used to implement Geometry Callback R*Tree filtering.

6.11.2.3 gaiaBuildMbr()

```
GAIA GEO_DECLARE void gaiaBuildMbr (
    double x1,
    double y1,
    double x2,
    double y2,
    int srid,
    unsigned char ** result,
    int * size )
```

Creates a BLOB-Geometry representing an Envelope [MBR].

Parameters

<i>x1</i>	first X coordinate.
<i>y1</i>	first Y coordinate.
<i>x2</i>	second X coordinate.
<i>y2</i>	second Y coordinate.
<i>srid</i>	the SRID associated to the Envelope
<i>result</i>	on completion will contain a pointer to newly created BLOB-Geometry
<i>size</i>	on completion this variable will contain the BLOB's size (in bytes)

See also

[gaiaBuildCircleMbr](#)

Note

[XY] coords must define two extreme Points identifying a diagonal of the MBR [Envelope]
no special order is required for coords: MAX / MIN values will be internally arranged as appropriate.

6.11.2.4 gaiaFromSpatialLiteBlobMbr()

```
GAIGEO_DECLARE gaiaGeomCollPtr gaiaFromSpatialLiteBlobMbr (
    const unsigned char * blob,
    unsigned int size )
```

Creates a Geometry object corresponding to the Envelope [MBR] for a BLOB-Geometry.

Parameters

<i>blob</i>	pointer to BLOB-Geometry
<i>size</i>	the BLOB's size (in bytes)

Returns

the pointer to the newly created Geometry object: NULL on failure

See also

[gaiaFreeGeomColl](#)

Note

you are responsible to destroy (before or after) any allocated Geometry, unless you've passed ownership of the Geometry object to some further object: in this case destroying the higher order object will implicitly destroy any contained child object.

6.11.2.5 `gaiaGetMbrMaxX()`

```
GAIA GEO_DECLARE int gaiaGetMbrMaxX (
    const unsigned char * blob,
    unsigned int size,
    double * maxx )
```

Retrieves the MBR (MaxX) from a BLOB-Geometry object.

Parameters

<i>blob</i>	pointer to BLOB-Geometry.
<i>size</i>	the BLOB's size (in bytes).
<i>maxx</i>	on completion this variable will contain the MBR MaxX coordinate.

Returns

0 on failure: any other value on success.

See also

[gaiaGetMbrMinX](#), [gaiaGetMbrMinY](#), [gaiaGetMbrMaxY](#)

6.11.2.6 `gaiaGetMbrMaxY()`

```
GAIA GEO_DECLARE int gaiaGetMbrMaxY (
    const unsigned char * blob,
    unsigned int size,
    double * maxy )
```

Retrieves the MBR (MaxY) from a BLOB-Geometry object.

Parameters

<i>blob</i>	pointer to BLOB-Geometry.
<i>size</i>	the BLOB's size (in bytes).
<i>maxy</i>	on completion this variable will contain the MBR MaxY coordinate.

Returns

0 on failure: any other value on success.

See also

[gaiaGetMbrMinX](#), [gaiaGetMbrMaxX](#), [gaiaGetMbrMinY](#)

6.11.2.7 `gaiaGetMbrMinX()`

```
GAIA_GEO_DECLARE int gaiaGetMbrMinX (
    const unsigned char * blob,
    unsigned int size,
    double * minx )
```

Retrieves the MBR (MinX) from a BLOB-Geometry object.

Parameters

<i>blob</i>	pointer to BLOB-Geometry.
<i>size</i>	the BLOB's size (in bytes).
<i>minx</i>	on completion this variable will contain the MBR MinX coordinate.

Returns

0 on failure: any other value on success.

See also

[gaiaGetMbrMaxX](#), [gaiaGetMbrMinY](#), [gaiaGetMbrMaxY](#)

6.11.2.8 `gaiaGetMbrMinY()`

```
GAIA_GEO_DECLARE int gaiaGetMbrMinY (
    const unsigned char * blob,
    unsigned int size,
    double * miny )
```

Retrieves the MBR (MinY) from a BLOB-Geometry object.

Parameters

<i>blob</i>	pointer to BLOB-Geometry.
<i>size</i>	the BLOB's size (in bytes).
<i>miny</i>	on completion this variable will contain the MBR MinY coordinate.

Returns

0 on failure: any other value on success.

See also

[gaiaGetMbrMinX](#), [gaiaGetMbrMaxX](#), [gaiaGetMbrMaxY](#)

6.11.2.9 gaiaMbrGeometry()

```
GAIAGEO_DECLARE void gaiaMbrGeometry (
    gaiaGeomCollPtr geom )
```

Updates the actual MBR for a Geometry object.

Parameters

<i>geom</i>	pointer to the Geometry object
-------------	--------------------------------

6.11.2.10 gaiaMbrLinestring()

```
GAIAGEO_DECLARE void gaiaMbrLinestring (
    gaiaLinestringPtr line )
```

Updates the actual MBR for a Linestring object.

Parameters

<i>line</i>	pointer to the Linestring object
-------------	----------------------------------

6.11.2.11 gaiaMbrPolygon()

```
GAIAGEO_DECLARE void gaiaMbrPolygon (
    gaiaPolygonPtr polyg )
```

Updates the actual MBR for a Polygon object.

Parameters

<i>polyg</i>	pointer to the Polygon object
--------------	-------------------------------

6.11.2.12 gaiaMbrRing()

```
GAIAGEO_DECLARE void gaiaMbrRing (
    gaiaRingPtr rng )
```

Updates the actual MBR for a Ring object.

Parameters

<i>rng</i>	pointer to the Ring object
------------	----------------------------

6.11.2.13 gaiaMbrsContains()

```
GAIA GEO_DECLARE int gaiaMbrsContains (
    gaiaGeomCollPtr mbr1,
    gaiaGeomCollPtr mbr2 )
```

MBRs comparison: Contains.

Parameters

<i>mbr1</i>	pointer to first Geometry object.
<i>mbr2</i>	pointer to second Geometry object.

Returns

0 if false; any other value if *mbr1* spatially *contains* *mbr2*

See also

[gaiaMbrsDisjoint](#), [gaiaMbrsEqual](#), [gaiaMbrsIntersects](#), [gaiaMbrsOverlaps](#), [gaiaMbrsTouches](#), [gaiaMbrsWithin](#)

6.11.2.14 gaiaMbrsDisjoint()

```
GAIA GEO_DECLARE int gaiaMbrsDisjoint (
    gaiaGeomCollPtr mbr1,
    gaiaGeomCollPtr mbr2 )
```

MBRs comparison: Disjoint.

Parameters

<i>mbr1</i>	pointer to first Geometry object.
<i>mbr2</i>	pointer to second Geometry object.

Returns

0 if false; any other value if *mbr1* and *mbr2* are spatially *disjoint*

See also

[gaiaMbrsContains](#), [gaiaMbrsEqual](#), [gaiaMbrsIntersects](#), [gaiaMbrsOverlaps](#), [gaiaMbrsTouches](#), [gaiaMbrsWithin](#)

6.11.2.15 `gaiaMbrsEqual()`

```
GAIA GEO_DECLARE int gaiaMbrsEqual (
    gaiaGeomCollPtr mbr1,
    gaiaGeomCollPtr mbr2 )
```

MBRs comparison: Equal.

Parameters

<i>mbr1</i>	pointer to first Geometry object.
<i>mbr2</i>	pointer to second Geometry object.

Returns

0 if false; any other value if *mbr1* and *mbr2* are spatially *equal*

See also

[gaiaMbrsContains](#), [gaiaMbrsDisjoint](#), [gaiaMbrsIntersects](#), [gaiaMbrsOverlaps](#), [gaiaMbrsTouches](#), [gaiaMbrsWithin](#)

6.11.2.16 `gaiaMbrsIntersects()`

```
GAIA GEO_DECLARE int gaiaMbrsIntersects (
    gaiaGeomCollPtr mbr1,
    gaiaGeomCollPtr mbr2 )
```

MBRs comparison: Intersects.

Parameters

<i>mbr1</i>	pointer to first Geometry object.
<i>mbr2</i>	pointer to second Geometry object.

Returns

0 if false; any other value if *mbr1* and *mbr2* spatially *intersect*

See also

[gaiaMbrsContains](#), [gaiaMbrsDisjoint](#), [gaiaMbrsEqual](#), [gaiaMbrsOverlaps](#), [gaiaMbrsTouches](#), [gaiaMbrsWithin](#)

6.11.2.17 `gaiaMbrsOverlaps()`

```
GAIA GEO_DECLARE int gaiaMbrsOverlaps (
    gaiaGeomCollPtr mbr1,
    gaiaGeomCollPtr mbr2 )
```

MBRs comparison: Overlaps.

Parameters

<i>mbr1</i>	pointer to first Geometry object.
<i>mbr2</i>	pointer to second Geometry object.

Returns

0 if false; any other value if *mbr1* and *mbr2* spatially *overlap*

See also

[gaiaMbrsContains](#), [gaiaMbrsDisjoint](#), [gaiaMbrsEqual](#), [gaiaMbrsIntersects](#), [gaiaMbrsTouches](#), [gaiaMbrsWithin](#)

6.11.2.18 `gaiaMbrsTouches()`

```
GAIA GEO_DECLARE int gaiaMbrsTouches (
    gaiaGeomCollPtr mbr1,
    gaiaGeomCollPtr mbr2 )
```

MBRs comparison: Touches.

Parameters

<i>mbr1</i>	pointer to first Geometry object.
<i>mbr2</i>	pointer to second Geometry object.

Returns

0 if false; any other value if *mbr1* and *mbr2* spatially *touche*

See also

[gaiaMbrsContains](#), [gaiaMbrsDisjoint](#), [gaiaMbrsEqual](#), [gaiaMbrsIntersects](#), [gaiaMbrsOverlaps](#), [gaiaMbrsWithin](#)

6.11.2.19 gaiaMbrsWithin()

```
GAIA GEO_DECLARE int gaiaMbrsWithin (
    gaiaGeomCollPtr mbr1,
    gaiaGeomCollPtr mbr2 )
```

MBRs comparison: Within.

Parameters

<i>mbr1</i>	pointer to first Geometry object.
<i>mbr2</i>	pointer to second Geometry object.

Returns

0 if false; any other value if mbr1 is spatially *within* mbr2

See also

[gaiaMbrsContains](#), [gaiaMbrsDisjoint](#), [gaiaMbrsEqual](#), [gaiaMbrsIntersects](#), [gaiaMbrsOverlaps](#), [gaiaMbrsTouches](#)

6.11.2.20 gaiaMRangeGeometry()

```
GAIA GEO_DECLARE void gaiaMRangeGeometry (
    gaiaGeomCollPtr geom,
    double * min,
    double * max )
```

Computes the Z-Range for a Geometry object.

Parameters

<i>geom</i>	pointer to the Geometry object
<i>min</i>	on completion this variable will contain the min M value found
<i>max</i>	on completion this variable will contain the max M value found

See also

[gaiaMRangeGeometryEx](#)

Note

if the Geometry has XY or XYZ dims, the M-Range is meaningless

6.11.2.21 gaiaMRangeGeometryEx()

```
GAIA GEO_DECLARE void gaiaMRangeGeometryEx (
    gaiaGeomCollPtr geom,
    double nodata,
    double * min,
    double * max )
```

Computes the Z-Range for a Geometry object (NODATA flavor)

Parameters

<i>geom</i>	pointer to the Geometry object
<i>nodata</i>	the NODATA value
<i>min</i>	on completion this variable will contain the min M value found
<i>max</i>	on completion this variable will contain the max M value found

See also

[gaiaMRangeGeometryEx](#)

Note

if the Geometry has XY or XYZ dims, the M-Range is meaningless

6.11.2.22 **gaiaMRangeLinestring()**

```
GAIAGEO_DECLARE void gaiaMRangeLinestring (
    gaiaLinestringPtr line,
    double * min,
    double * max )
```

Computes the M-Range for a Linestring object.

Parameters

<i>line</i>	pointer to the Linestring object
<i>min</i>	on completion this variable will contain the min M value found
<i>max</i>	on completion this variable will contain the max M value found

See also

[gaiaMRangeLinestringEx](#)

Note

if the Linestring has XY or XYZ dims, the M-Range is meaningless

6.11.2.23 **gaiaMRangeLinestringEx()**

```
GAIAGEO_DECLARE void gaiaMRangeLinestringEx (
    gaiaLinestringPtr line,
    double nodata,
    double * min,
    double * max )
```

Computes the M-Range for a Linestring object (NODATA flavor)

Parameters

<i>line</i>	pointer to the Linestring object
<i>nodata</i>	the NODATA value
<i>min</i>	on completion this variable will contain the min M value found
<i>max</i>	on completion this variable will contain the max M value found

See also

[gaiaMRangeLinestring](#)

Note

if the Linestring has XY or XYZ dims, the M-Range is meaningless

6.11.2.24 gaiaMRangePolygon()

```
GAIAGEO_DECLARE void gaiaMRangePolygon (
    gaiaPolygonPtr polyg,
    double * min,
    double * max )
```

Computes the M-Range for a Polygon object.

Parameters

<i>polyg</i>	pointer to the Polygon object
<i>min</i>	on completion this variable will contain the min M value found
<i>max</i>	on completion this variable will contain the max M value found

See also

[gaiaMRangePolygonEx](#)

Note

if the Polygon has XY or XYZ dims, the M-Range is meaningless

6.11.2.25 gaiaMRangePolygonEx()

```
GAIAGEO_DECLARE void gaiaMRangePolygonEx (
    gaiaPolygonPtr polyg,
    double nodata,
    double * min,
    double * max )
```

Computes the M-Range for a Polygon object (NODATA flavor)

Parameters

<i>polyg</i>	pointer to the Polygon object
<i>nodata</i>	the NODATA value
<i>min</i>	on completion this variable will contain the min M value found
<i>max</i>	on completion this variable will contain the max M value found

See also

[gaiaMRangePolygon](#)

Note

if the Polygon has XY or XYZ dims, the M-Range is meaningless

6.11.2.26 gaiaMRangeRing()

```
GAIA GEO_DECLARE void gaiaMRangeRing (
    gaiaRingPtr rng,
    double * min,
    double * max )
```

Computes the M-Range for a Ring object.

Parameters

<i>rng</i>	pointer to the Ring object
<i>min</i>	on completion this variable will contain the min M value found
<i>max</i>	on completion this variable will contain the max M value found

See also

[gaiaMRangeRingEx](#)

Note

if the Ring has XY or XYZ dims, the M-Range is meaningless

6.11.2.27 gaiaMRangeRingEx()

```
GAIA GEO_DECLARE void gaiaMRangeRingEx (
    gaiaRingPtr rng,
    double nodata,
    double * min,
    double * max )
```

Computes the M-Range for a Ring object (NODATA flavor)

Parameters

<i>rng</i>	pointer to the Ring object
<i>nodata</i>	the NODATA value
<i>min</i>	on completion this variable will contain the min M value found
<i>max</i>	on completion this variable will contain the max M value found

See also

[gaiaMRangeRing](#)

Note

if the Ring has XY or XYZ dims, the M-Range is meaningless

6.11.2.28 gaiaParseFilterMbr()

```
GAIAGEO_DECLARE int gaiaParseFilterMbr (
    unsigned char * result,
    int size,
    double * minx,
    double * miny,
    double * maxx,
    double * maxy,
    int * mode )
```

Creates a BLOB-FilterMBR.

Parameters

<i>result</i>	pointer to BLOB-FilterMBR [previously created by gaiaBuildFilterMbr] BLOB-Geometry
<i>size</i>	BLOB's size (in bytes)
<i>minx</i>	on completion this variable will contain the MBR MinX coord.
<i>miny</i>	on completion this variable will contain the MBR MinY coord.
<i>maxx</i>	on completion this variable will contain the MBR MinY coord.
<i>maxy</i>	on completion this variable will contain the MBR MaxY coord.
<i>mode</i>	on completion this variable will contain the FilterMBR mode.

See also

[gaiaBuildFilterMbr](#)

Remarks

internally used to implement Geometry Callback R*Tree filtering.

6.11.2.29 `gaiaZRangeGeometry()`

```
GAIA GEO_DECLARE void gaiaZRangeGeometry (
    gaiaGeomCollPtr geom,
    double * min,
    double * max )
```

Computes the Z-Range for a Geometry object.

Parameters

<i>geom</i>	pointer to the Geometry object
<i>min</i>	on completion this variable will contain the min Z value found
<i>max</i>	on completion this variable will contain the max Z value found

See also

[gaiaZRangeGeometryEx](#)

Note

if the Geometry has XY or XYM dims, the Z-Range is meaningless

6.11.2.30 `gaiaZRangeGeometryEx()`

```
GAIA GEO_DECLARE void gaiaZRangeGeometryEx (
    gaiaGeomCollPtr geom,
    double nodata,
    double * min,
    double * max )
```

Computes the Z-Range for a Geometry object (NODATA flavor)

Parameters

<i>geom</i>	pointer to the Geometry object
<i>nodata</i>	the NODATA value
<i>min</i>	on completion this variable will contain the min Z value found
<i>max</i>	on completion this variable will contain the max Z value found

See also

[gaiaZRangeGeometry](#)

Note

if the Geometry has XY or XYM dims, the Z-Range is meaningless

6.11.2.31 `gaiaZRangeLinestring()`

```
GAIA GEO_DECLARE void gaiaZRangeLinestring (
    gaiaLinestringPtr line,
    double * min,
    double * max )
```

Computes the Z-Range for a Linestring object.

Parameters

<i>line</i>	pointer to the Linestring object
<i>min</i>	on completion this variable will contain the min Z value found
<i>max</i>	on completion this variable will contain the max Z value found

See also

[gaiaZRangeLinestringEx](#)

Note

if the Linestring has XY or XYM dims, the Z-Range is meaningless

6.11.2.32 `gaiaZRangeLinestringEx()`

```
GAIA GEO_DECLARE void gaiaZRangeLinestringEx (
    gaiaLinestringPtr line,
    double nodata,
    double * min,
    double * max )
```

Computes the Z-Range for a Linestring object (NODATA flavor)

Parameters

<i>line</i>	pointer to the Linestring object
<i>nodata</i>	the NODATA value
<i>min</i>	on completion this variable will contain the min Z value found
<i>max</i>	on completion this variable will contain the max Z value found

See also

[gaiaZRangeLinestring](#)

Note

if the Linestring has XY or XYM dims, the Z-Range is meaningless

6.11.2.33 `gaiaZRangePolygon()`

```
GAIA GEO_DECLARE void gaiaZRangePolygon (
    gaiaPolygonPtr polyg,
    double * min,
    double * max )
```

Computes the Z-Range for a Polygon object.

Parameters

<i>polyg</i>	pointer to the Polygon object
<i>min</i>	on completion this variable will contain the min Z value found
<i>max</i>	on completion this variable will contain the max Z value found

See also

[gaiaZRangePolygonEx](#)

Note

if the Polygon has XY or XYM dims, the Z-Range is meaningless

6.11.2.34 `gaiaZRangePolygonEx()`

```
GAIA GEO_DECLARE void gaiaZRangePolygonEx (
    gaiaPolygonPtr polyg,
    double nodata,
    double * min,
    double * max )
```

Computes the Z-Range for a Polygon object (NODATA flavor)

Parameters

<i>polyg</i>	pointer to the Polygon object
<i>nodata</i>	the NODATA value
<i>min</i>	on completion this variable will contain the min Z value found
<i>max</i>	on completion this variable will contain the max Z value found

See also

[gaiaZRangePolygon](#)

Note

if the Polygon has XY or XYM dims, the Z-Range is meaningless

6.11.2.35 `gaiaZRangeRing()`

```
GAIA GEO_DECLARE void gaiaZRangeRing (
    gaiaRingPtr rng,
    double * min,
    double * max )
```

Computes the Z-Range for a Ring object.

Parameters

<i>rng</i>	pointer to the Ring object
<i>min</i>	on completion this variable will contain the min Z value found
<i>max</i>	on completion this variable will contain the max Z value found

See also

[gaiaZRangeRingEx](#)

Note

if the Ring has XY or XYM dims, the Z-Range is meaningless

6.11.2.36 `gaiaZRangeRingEx()`

```
GAIA GEO_DECLARE void gaiaZRangeRingEx (
    gaiaRingPtr rng,
    double nodata,
    double * min,
    double * max )
```

Computes the Z-Range for a Ring object (NODATA flavor)

Parameters

<i>rng</i>	pointer to the Ring object
<i>nodata</i>	the NODATA value
<i>min</i>	on completion this variable will contain the min Z value found
<i>max</i>	on completion this variable will contain the max Z value found

See also

[gaiaZRangeRing](#)

Note

if the Ring has XY or XYM dims, the Z-Range is meaningless

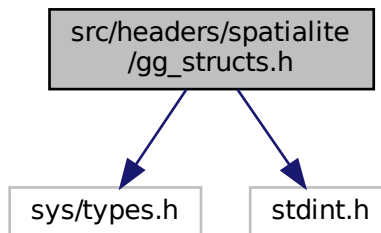
6.12 src/headers/spatialite/gg_structs.h File Reference

Geometry structures.

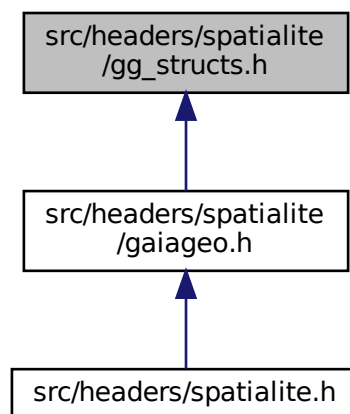
```
#include <sys/types.h>
```

```
#include <stdint.h>
```

Include dependency graph for gg_structs.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [gaiaPointStruct](#)
Container for OGC POINT Geometry.
- struct [gaiaDynamicLineStruct](#)
Container for dynamically growing line/ring.
- struct [gaiaLinestringStruct](#)

- Container for OGC LINESTRING Geometry.*

 - struct [gaiaRingStruct](#)
- Container for OGC RING Geometry.*

 - struct [gaiaPolygonStruct](#)
- Container for OGC POLYGON Geometry.*

 - struct [gaiaGeomCollStruct](#)
- Container for OGC GEOMETRYCOLLECTION Geometry.*

 - struct [gaiaPreRingStruct](#)
- Container similar to LINESTRING [internally used].*

 - struct [gaiaValueStruct](#)
- Container for variant (multi-type) value.*

 - struct [gaiaDbfFieldStruct](#)
- Container for DBF field.*

 - struct [gaiaDbfListStruct](#)
- Container for a list of DBF fields.*

 - struct [gaiaMemFileStruct](#)
- A Memory based File.*

 - struct [gaiaDbfStruct](#)
- Container for DBF file handling.*

 - struct [gaiaShapefileStruct](#)
- Container for SHP file handling.*

 - struct [gaiaOutBufferStruct](#)
- Container for dynamically growing output buffer.*

 - struct [vrttxt_line](#)
- Container for Virtual Text record (line)*

 - struct [vrttxt_row](#)
- Container for Virtual Text record (line) offsets.*

 - struct [vrttxt_row_block](#)
- Container for Virtual Text block of records.*

 - struct [vrttxt_column_header](#)
- Container for Virtual Text column (field) header.*

 - struct [vrttxt_reader](#)
- Container for Virtual Text file handling.*

 - struct [gaiaLayerExtentInfos](#)
- Layer Extent infos.*

 - struct [gaiaLayerAuthInfos](#)
- Layer Auth infos.*

 - struct [gaiaAttributeFieldMaxSizeInfos](#)
- Attribute/Field MaxSize/Length infos.*

 - struct [gaiaAttributeFieldIntRangeInfos](#)
- Attribute/Field Integer range infos.*

 - struct [gaiaAttributeFieldDoubleRangeInfos](#)
- Attribute/Field Double range infos.*

 - struct [gaiaLayerAttributeFieldInfos](#)
- LayerAttributeField infos.*

 - struct [gaiaVectorLayerItem](#)
- Vector Layer item.*

 - struct [gaiaVectorLayersListStr](#)
- Container for Vector Layers List.*

 - struct [gaiaProjAreaStr](#)
- BBOX corresponding to PROJ.6 AREA.*

Macros

- `#define gaia_off_t off_t`
- `#define gaia_fseek fseeko`
- `#define VRTTXT_FIELDS_MAX 65535`
Virtual Text driver: MAX number of fields.
- `#define VRTTXT_BLOCK_MAX 65535`
Virtual Text driver: MAX block size (in bytes)
- `#define VRTTXT_TEXT 1`
Virtual Text driver: TEXT value.
- `#define VRTTXT_INTEGER 2`
Virtual Text driver: INTEGER value.
- `#define VRTTXT_DOUBLE 3`
Virtual Text driver: DOUBLE value.
- `#define VRTTXT_NULL 4`
Virtual Text driver: NULL value.

Typedefs

- `typedef struct gaiaPointStruct gaiaPoint`
Container for OGC POINT Geometry.
- `typedef gaiaPoint * gaiaPointPtr`
Typedef for OGC POINT structure.
- `typedef struct gaiaDynamicLineStruct gaiaDynamicLine`
Container for dynamically growing line/ring.
- `typedef gaiaDynamicLine * gaiaDynamicLinePtr`
Typedef for dynamically growing line/ring structure.
- `typedef struct gaiaLinestringStruct gaiaLinestring`
Container for OGC LINESTRING Geometry.
- `typedef gaiaLinestring * gaiaLinestringPtr`
Typedef for OGC LINESTRING structure.
- `typedef struct gaiaRingStruct gaiaRing`
Container for OGC RING Geometry.
- `typedef gaiaRing * gaiaRingPtr`
Typedef for OGC RING structure.
- `typedef struct gaiaPolygonStruct gaiaPolygon`
Container for OGC POLYGON Geometry.
- `typedef gaiaPolygon * gaiaPolygonPtr`
Typedef for OGC POLYGON structure.
- `typedef struct gaiaGeomCollStruct gaiaGeomColl`
Container for OGC GEOMETRYCOLLECTION Geometry.
- `typedef gaiaGeomColl * gaiaGeomCollPtr`
Typedef for OGC GEOMETRYCOLLECTION structure.
- `typedef struct gaiaPreRingStruct gaiaPreRing`
Container similar to LINESTRING [internally used].
- `typedef gaiaPreRing * gaiaPreRingPtr`
Typedef for gaiaPreRing structure.
- `typedef struct gaiaValueStruct gaiaValue`
Container for variant (multi-type) value.
- `typedef gaiaValue * gaiaValuePtr`

- Typedef for variant (multi-type) value structure.*

 - typedef struct [gaiaDbfFieldStruct](#) [gaiaDbfField](#)
 - Container for DBF field.*
 - typedef [gaiaDbfField](#) * [gaiaDbfFieldPtr](#)
 - Typedef for DBF field structure.*
 - typedef struct [gaiaDbfListStruct](#) [gaiaDbfList](#)
 - Container for a list of DBF fields.*
 - typedef [gaiaDbfList](#) * [gaiaDbfListPtr](#)
 - Typedef for a list of DBF fields.*
 - typedef struct [gaiaMemFileStruct](#) [gaiaMemFile](#)
 - A Memory based File.*
 - typedef [gaiaMemFile](#) * [gaiaMemFilePtr](#)
 - Typedef for Memory File structure.*
 - typedef struct [gaiaDbfStruct](#) [gaiaDbf](#)
 - Container for DBF file handling.*
 - typedef [gaiaDbf](#) * [gaiaDbfPtr](#)
 - Typedef for DBF file handler structure.*
 - typedef struct [gaiaShapefileStruct](#) [gaiaShapefile](#)
 - Container for SHP file handling.*
 - typedef [gaiaShapefile](#) * [gaiaShapefilePtr](#)
 - Typedef for SHP file handler structure.*
 - typedef struct [gaiaOutBufferStruct](#) [gaiaOutBuffer](#)
 - Container for dynamically growing output buffer.*
 - typedef [gaiaOutBuffer](#) * [gaiaOutBufferPtr](#)
 - Typedef for dynamically growing output buffer structure.*
 - typedef struct [vrttxt_reader](#) [gaiaTextReader](#)
 - Container for Virtual Text file handling.*
 - typedef [gaiaTextReader](#) * [gaiaTextReaderPtr](#)
 - Typedef for Virtual Text file handling structure.*
 - typedef struct [gaiaLayerExtentInfos](#) [gaiaLayerExtent](#)
 - Layer Extent infos.*
 - typedef [gaiaLayerExtent](#) * [gaiaLayerExtentPtr](#)
 - Typedef for Layer Extent infos.*
 - typedef struct [gaiaLayerAuthInfos](#) [gaiaLayerAuth](#)
 - Layer Auth infos.*
 - typedef [gaiaLayerAuth](#) * [gaiaLayerAuthPtr](#)
 - Typedef for Layer Auth infos.*
 - typedef struct [gaiaAttributeFieldMaxSizeInfos](#) [gaiaAttributeFieldMaxSize](#)
 - Attribute/Field MaxSize/Length infos.*
 - typedef [gaiaAttributeFieldMaxSize](#) * [gaiaAttributeFieldMaxSizePtr](#)
 - Typedef for Attribute/Field MaxSize/Length infos.*
 - typedef struct [gaiaAttributeFieldIntRangeInfos](#) [gaiaAttributeFieldIntRange](#)
 - Attribute/Field Integer range infos.*
 - typedef [gaiaAttributeFieldIntRange](#) * [gaiaAttributeFieldIntRangePtr](#)
 - Typedef for Attribute/Field Integer range infos.*
 - typedef struct [gaiaAttributeFieldDoubleRangeInfos](#) [gaiaAttributeFieldDoubleRange](#)
 - Attribute/Field Double range infos.*
 - typedef [gaiaAttributeFieldDoubleRange](#) * [gaiaAttributeFieldDoubleRangePtr](#)
 - Typedef for Attribute/Field Double range infos.*
 - typedef struct [gaiaLayerAttributeFieldInfos](#) [gaiaLayerAttributeField](#)
 - LayerAttributeField infos.*

- typedef [gaiaLayerAttributeField](#) * [gaiaLayerAttributeFieldPtr](#)
Typedef for Layer AttributeField infos.
- typedef struct [gaiaVectorLayerItem](#) [gaiaVectorLayer](#)
Vector Layer item.
- typedef [gaiaVectorLayer](#) * [gaiaVectorLayerPtr](#)
Typedef for Vector Layer item.
- typedef struct [gaiaVectorLayersListStr](#) [gaiaVectorLayersList](#)
Container for Vector Layers List.
- typedef [gaiaVectorLayersList](#) * [gaiaVectorLayersListPtr](#)
Typedef for Vector Layers List.
- typedef struct [gaiaProjAreaStr](#) [gaiaProjArea](#)
BBOX corresponding to PROJ.6 AREA.
- typedef [gaiaProjArea](#) * [gaiaProjAreaPtr](#)
Typedef for BBOX corresponding to PROJ.6 AREA.

6.12.1 Detailed Description

Geometry structures.

6.12.2 Typedef Documentation

6.12.2.1 [gaiaAttributeFieldDoubleRangePtr](#)

```
typedef gaiaAttributeFieldDoubleRange* gaiaAttributeFieldDoubleRangePtr
```

Typedef for Attribute/Field Double range infos.

See also

[gaiaAttributeFieldDoubleRange](#)

6.12.2.2 [gaiaAttributeFieldIntRangePtr](#)

```
typedef gaiaAttributeFieldIntRange* gaiaAttributeFieldIntRangePtr
```

Typedef for Attribute/Field Integer range infos.

See also

[gaiaAttributeFieldIntRange](#)

6.12.2.3 `gaiaAttributeFieldMaxSizePtr`

```
typedef gaiaAttributeFieldMaxSize\* gaiaAttributeFieldMaxSizePtr
```

Typedef for Attribute/Field MaxSize/Length infos.

See also

[gaiaAttributeFieldMaxSize](#)

6.12.2.4 `gaiaDbfListPtr`

```
typedef gaiaDbfList\* gaiaDbfListPtr
```

Typedef for a list of DBF fields.

See also

[gaiaDbfList](#)

6.12.2.5 `gaiaDbfPtr`

```
typedef gaiaDbf\* gaiaDbfPtr
```

Typedef for DBF file handler structure.

See also

[gaiaDbf](#)

6.12.2.6 `gaiaDynamicLinePtr`

```
typedef gaiaDynamicLine\* gaiaDynamicLinePtr
```

Typedef for dynamically growing line/ring structure.

See also

[gaiaDynamicLine](#)

6.12.2.7 `gaiaGeomCollPtr`

```
typedef gaiaGeomColl\* gaiaGeomCollPtr
```

Typedef for OGC GEOMETRYCOLLECTION structure.

See also

[gaiaGeomCool](#)

6.12.2.8 `gaiaLayerAttributeFieldPtr`

```
typedef gaiaLayerAttributeField\* gaiaLayerAttributeFieldPtr
```

Typedef for Layer AttributeField infos.

See also

[gaiaLayerAttributeField](#)

6.12.2.9 `gaiaLayerAuthPtr`

```
typedef gaiaLayerAuth\* gaiaLayerAuthPtr
```

Typedef for Layer Auth infos.

See also

[gaiaLayerAuth](#)

6.12.2.10 `gaiaLayerExtentPtr`

```
typedef gaiaLayerExtent\* gaiaLayerExtentPtr
```

Typedef for Layer Extent infos.

See also

[gaiaLayerExtent](#)

6.12.2.11 `gaiaLinestringPtr`

```
typedef gaiaLinestring\* gaiaLinestringPtr
```

Typedef for OGC LINESTRING structure.

See also

[gaiaLinestring](#)

6.12.2.12 `gaiaMemFilePtr`

```
typedef gaiaMemFile\* gaiaMemFilePtr
```

Typedef for Memory File structure.

See also

[gaiaMemFile](#)

6.12.2.13 `gaiaOutBufferPtr`

```
typedef gaiaOutBuffer\* gaiaOutBufferPtr
```

Typedef for dynamically growing output buffer structure.

See also

[gaiaOutBuffer](#)

6.12.2.14 `gaiaPointPtr`

```
typedef gaiaPoint\* gaiaPointPtr
```

Typedef for OGC POINT structure.

See also

[gaiaPoint](#)

6.12.2.15 `gaiaPolygonPtr`

```
typedef gaiaPolygon\* gaiaPolygonPtr
```

Typedef for OGC POLYGON structure.

See also

[gaiaPolygon](#)

6.12.2.16 `gaiaPreRingPtr`

```
typedef gaiaPreRing\* gaiaPreRingPtr
```

Typedef for `gaiaPreRing` structure.

See also

[gaiaPreRing](#)

6.12.2.17 `gaiaProjAreaPtr`

```
typedef gaiaProjArea\* gaiaProjAreaPtr
```

Typedef for BBOX corresponding to PROJ.6 AREA.

See also

[gaiaProjArea](#)

6.12.2.18 `gaiaRingPtr`

```
typedef gaiaRing\* gaiaRingPtr
```

Typedef for OGC RING structure.

See also

[gaiaRing](#)

6.12.2.19 `gaiaShapefilePtr`

```
typedef gaiaShapefile\* gaiaShapefilePtr
```

Typedef for SHP file handler structure.

See also

[gaiaShapefile](#)

6.12.2.20 `gaiaTextReaderPtr`

```
typedef gaiaTextReader\* gaiaTextReaderPtr
```

Typedef for Virtual Text file handling structure.

See also

[gaiaTextReader](#)

6.12.2.21 `gaiaVectorLayerPtr`

```
typedef gaiaVectorLayer\* gaiaVectorLayerPtr
```

Typedef for Vector Layer item.

See also

[gaiaVectorLayer](#)

6.12.2.22 `gaiaVectorLayersListPtr`

```
typedef gaiaVectorLayersList\* gaiaVectorLayersListPtr
```

Typedef for Vector Layers List.

See also

[gaiaVectorLayersList](#)

6.13 `src/headers/spatialite/gg_wfs.h` File Reference

WFS support.

Typedefs

- typedef struct gaia_wfs_catalog **gaiaWFScatalog**
- typedef gaiaWFScatalog * **gaiaWFScatalogPtr**
- typedef struct gaia_wfs_item **gaiaWFSitem**
- typedef gaiaWFSitem * **gaiaWFSitemPtr**
- typedef struct gaia_wfs_schema **gaiaWFSschema**
- typedef gaiaWFSschema * **gaiaWFSschemaPtr**
- typedef struct gaia_wfs_column **gaiaWFScolumn**
- typedef gaiaWFScolumn * **gaiaWFScolumnPtr**

Functions

- SPATIALITE_DECLARE int [load_from_wfs](#) (sqlite3 *sqlite, const char *path_or_url, const char *alt_↵ describe_uri, const char *layer_name, int swap_axes, const char *table, const char *pk_column_name, int spatial_index, int *rows, char **err_msg, void(*progress_callback)(int, void *), void *callback_ptr)
Loads data from some WFS source.
- SPATIALITE_DECLARE int [load_from_wfs_paged](#) (sqlite3 *sqlite, const char *path_or_url, const char *alt_↵ _describe_uri, const char *layer_name, int swap_axes, const char *table, const char *pk_column_name, int spatial_index, int page_size, int *rows, char **err_msg, void(*progress_callback)(int, void *), void *callback_ptr)
Loads data from some WFS source (using WFS paging)
- SPATIALITE_DECLARE int [load_from_wfs_paged_ex](#) (sqlite3 *sqlite, const char *wfs_version, const char *path_or_url, const char *alt_describe_uri, const char *layer_name, int swap_axes, const char *table, const char *pk_column_name, int spatial_index, int page_size, int *rows, char **err_msg, void(*progress_↵ callback)(int, void *), void *callback_ptr)
Loads data from some WFS source (using WFS paging) - Extended.
- SPATIALITE_DECLARE gaiaWFScatalogPtr [create_wfs_catalog](#) (const char *path_or_url, char **err_msg)
Creates a Catalog for some WFS service.
- SPATIALITE_DECLARE void [destroy_wfs_catalog](#) (gaiaWFScatalogPtr handle)
Destroys a WFS-Catalog object freeing any allocated resource.
- SPATIALITE_DECLARE const char * [get_wfs_version](#) (gaiaWFScatalogPtr handle)
Return the WFS-Version string as reported by GetCapabilities.
- SPATIALITE_DECLARE const char * [get_wfs_base_request_url](#) (gaiaWFScatalogPtr handle)
Return the base URL for any WFS-GetFeature call.
- SPATIALITE_DECLARE const char * [get_wfs_base_describe_url](#) (gaiaWFScatalogPtr handle)
Return the base URL for any WFS-DescribeFeatureType call.
- SPATIALITE_DECLARE char * [get_wfs_request_url](#) (gaiaWFScatalogPtr handle, const char *name, const char *version, int srid, int max_features)
Return a GetFeature URL (GET)
- SPATIALITE_DECLARE char * [get_wfs_describe_url](#) (gaiaWFScatalogPtr handle, const char *name, const char *version)
Return a DescribeFeatureType URL (GET)
- SPATIALITE_DECLARE int [get_wfs_catalog_count](#) (gaiaWFScatalogPtr handle)
Return the total count of items (aka Layers) defined within a WFS-Catalog object.
- SPATIALITE_DECLARE gaiaWFSitemPtr [get_wfs_catalog_item](#) (gaiaWFScatalogPtr handle, int index)
Return the pointer to some specific Layer defined within a WFS-Catalog object.
- SPATIALITE_DECLARE const char * [get_wfs_item_name](#) (gaiaWFSitemPtr handle)
Return the name corresponding to some WFS-Item (aka Layer) object.
- SPATIALITE_DECLARE const char * [get_wfs_item_title](#) (gaiaWFSitemPtr handle)
Return the title corresponding to some WFS-Item (aka Layer) object.
- SPATIALITE_DECLARE const char * [get_wfs_item_abstract](#) (gaiaWFSitemPtr handle)

- Return the abstract corresponding to some WFS-Item (aka Layer) object.*
- SPATIALITE_DECLARE int [get_wfs_layer_srid_count](#) (gaiaWFSitemPtr handle)
- Return the total count of SRIDs supported by a WFS-Item object.*
- SPATIALITE_DECLARE int [get_wfs_layer_srid](#) (gaiaWFSitemPtr handle, int index)
- Return one of the SRIDs supported by a WFS-Item object.*
- SPATIALITE_DECLARE int [get_wfs_keyword_count](#) (gaiaWFSitemPtr handle)
- Return the total count of Keywords associated to a WFS-Item object.*
- SPATIALITE_DECLARE const char * [get_wfs_keyword](#) (gaiaWFSitemPtr handle, int index)
- Return one of the Keywords supported by a WFS-Item object.*
- SPATIALITE_DECLARE gaiaWFSschemaPtr [create_wfs_schema](#) (const char *path_or_url, const char *layer_name, char **err_msg)
- Creates a Schema representing some WFS Layer.*
- SPATIALITE_DECLARE void [destroy_wfs_schema](#) (gaiaWFSschemaPtr handle)
- Destroys a WFS-schema object freeing any allocated resource.*
- SPATIALITE_DECLARE int [get_wfs_schema_geometry_info](#) (gaiaWFSschemaPtr handle, const char **name, int *type, int *srid, int *dims, int *nullable)
- Return the infos describing some WFS-GeometryColumn object.*
- SPATIALITE_DECLARE int [get_wfs_schema_column_count](#) (gaiaWFSschemaPtr handle)
- Return the total count of items (aka Columns) defined within a WFS-Schema object.*
- SPATIALITE_DECLARE gaiaWFScolumnPtr [get_wfs_schema_column](#) (gaiaWFSschemaPtr handle, int index)
- Return the pointer to some specific Column defined within a WFS-Schema object.*
- SPATIALITE_DECLARE int [get_wfs_schema_column_info](#) (gaiaWFScolumnPtr handle, const char **name, int *type, int *nullable)
- Return the infos describing some WFS-Column object.*
- SPATIALITE_DECLARE void [reset_wfs_http_connection](#) (void)
- Resets the libxml2 "nano HTTP": useful when changing the HTTP_PROXY settings.*

6.13.1 Detailed Description

WFS support.

6.13.2 Function Documentation

6.13.2.1 [create_wfs_catalog\(\)](#)

```
SPATIALITE_DECLARE gaiaWFScatalogPtr create_wfs_catalog (
    const char * path_or_url,
    char ** err_msg )
```

Creates a Catalog for some WFS service.

Parameters

<i>path_or_url</i>	pointer to some WFS-GetCapabilities XML Document (could be a pathname or an URL).
<i>err_msg</i>	on completion will contain an error message (if any)

Returns

the pointer to the corresponding WFS-Catalog object: NULL on failure

See also

[destroy_wfs_catalog](#), [get_wfs_catalog_count](#), [get_wfs_catalog_item](#), [load_from_wfs](#), [reset_wfs_http_connection](#), [get_wfs_version](#)

Note

an eventual error message returned via `err_msg` requires to be deallocated by invoking `free()`.
you are responsible to destroy (before or after) any WFS-Catalog returned by [create_wfs_catalog\(\)](#).

6.13.2.2 create_wfs_schema()

```
SPATIALITE_DECLARE gaiaWFSschemaPtr create_wfs_schema (
    const char * path_or_url,
    const char * layer_name,
    char ** err_msg )
```

Creates a Schema representing some WFS Layer.

Parameters

<i>path_or_url</i>	pointer to some WFS-DescribeFeatureType XML Document (could be a pathname or an URL).
<i>err_msg</i>	on completion will contain an error message (if any)

Returns

the pointer to the corresponding WFS-Schema object: NULL on failure

See also

[destroy_wfs_schema](#), [get_wfs_schema_column_count](#), [get_wfs_schema_column_info](#), [get_wfs_schema_geometry_info](#)

Note

an eventual error message returned via `err_msg` requires to be deallocated by invoking `free()`.
you are responsible to destroy (before or after) any WFS-Schema returned by [create_wfs_schema\(\)](#).

6.13.2.3 destroy_wfs_catalog()

```
SPATIALITE_DECLARE void destroy_wfs_catalog (
    gaiaWFScatalogPtr handle )
```

Destroys a WFS-Catalog object freeing any allocated resource.

Parameters

<i>handle</i>	the pointer to a valid WFS-Catalog returned by a previous call to create_wfs_catalog()
---------------	--

See also[create_wfs_catalog](#)**6.13.2.4 destroy_wfs_schema()**

```
SPATIALITE_DECLARE void destroy_wfs_schema (  
    gaiaWFSschemaPtr handle )
```

Destroys a WFS-schema object freeing any allocated resource.

Parameters

<i>handle</i>	the pointer to a valid WFS-Catalog returned by a previous call to create_wfs_schema()
---------------	---

See also[create_wfs_schema](#)**6.13.2.5 get_wfs_base_describe_url()**

```
SPATIALITE_DECLARE const char* get_wfs_base_describe_url (  
    gaiaWFScatalogPtr handle )
```

Return the base URL for any WFS-DescribeFeatureType call.

Parameters

<i>handle</i>	the pointer to a valid WFS-Item returned by a previous call to get_wfs_catalog_item() .
---------------	---

Returns

the base URL for any WFS-DescribeFeatureType call: NULL is undefined

See also[create_wfs_catalog](#), [get_wfs_base_request_url](#), [get_wfs_describe_url](#)

6.13.2.6 `get_wfs_base_request_url()`

```
SPATIALITE_DECLARE const char* get_wfs_base_request_url (
    gaiaWFScatalogPtr handle )
```

Return the base URL for any WFS-GetFeature call.

Parameters

<i>handle</i>	the pointer to a valid WFS-Item returned by a previous call to get_wfs_catalog_item() .
---------------	---

Returns

the base URL for any WFS-GetFeature call: NULL is undefined

See also

[create_wfs_catalog](#), [get_wfs_base_describe_url](#), [get_wfs_request_url](#)

6.13.2.7 `get_wfs_catalog_count()`

```
SPATIALITE_DECLARE int get_wfs_catalog_count (
    gaiaWFScatalogPtr handle )
```

Return the total count of items (aka Layers) defined within a WFS-Catalog object.

Parameters

<i>handle</i>	the pointer to a valid WFS-Catalog returned by a previous call to create_wfs_catalog()
---------------	--

Returns

the total count of items (aka Layers) defined within a WFS-Catalog object: a negative number if the WFS-Catalog isn't valid

See also

[create_wfs_catalog](#), [get_wfs_catalog_item](#)

6.13.2.8 `get_wfs_catalog_item()`

```
SPATIALITE_DECLARE gaiaWFSitemPtr get_wfs_catalog_item (
    gaiaWFScatalogPtr handle,
    int index )
```

Return the pointer to some specific Layer defined within a WFS-Catalog object.

Parameters

<i>handle</i>	the pointer to a valid WFS-Catalog returned by a previous call to create_wfs_catalog()
<i>index</i>	the relative index identifying the required WFS-Layer (the first Item in the WFS-Catalog object has index ZERO).

Returns

the pointer to the required WFS-Layer object: NULL if the passed index isn't valid

See also

[create_wfs_catalog](#), [get_wfs_catalog_count](#), [get_wfs_item_name](#), [get_wfs_item_title](#), [get_wfs_item_abstract](#), [get_wfs_layer_srid_count](#), [get_wfs_layer_srid](#), [get_wfs_keyword_count](#), [get_wfs_keyword](#)

6.13.2.9 get_wfs_describe_url()

```
SPATIALITE_DECLARE char* get_wfs_describe_url (
    gaiaWFScatalogPtr handle,
    const char * name,
    const char * version )
```

Return a DescribeFeatureType URL (GET)

Parameters

<i>handle</i>	the pointer to a valid WFS-Item returned by a previous call to get_wfs_catalog_item() .
<i>name</i>	the NAME uniquely identifying the required WFS layer.
<i>version</i>	could be "1.0.0" or "1.1.0"; if NULL or invalid "1.1.0" will be assumed.

Returns

the DescribeFeatureType URL: NULL if any error is found.

See also

[get_wfs_base_describe_url](#), [get_wfs_request_url](#)

Note

you are responsible to destroy (before or after) any allocated memory returned by [get_wfs_describe_url\(\)](#).

6.13.2.10 get_wfs_item_abstract()

```
SPATIALITE_DECLARE const char* get_wfs_item_abstract (
    gaiaWFSitemPtr handle )
```

Return the abstract corresponding to some WFS-Item (aka Layer) object.

Parameters

<i>handle</i>	the pointer to a valid WFS-Item returned by a previous call to get_wfs_catalog_item() .
---------------	---

Returns

the abstract corresponding to the WFS-Layer object

See also

[get_wfs_item_name](#), [get_wfs_item_title](#), [get_wfs_layer_srid_count](#), [get_wfs_layer_srid](#), [get_wfs_keyword_count](#), [get_wfs_keyword](#)

6.13.2.11 get_wfs_item_name()

```
SPATIALITE_DECLARE const char* get_wfs_item_name (  
    gaiaWFSitemPtr handle )
```

Return the name corresponding to some WFS-Item (aka Layer) object.

Parameters

<i>handle</i>	the pointer to a valid WFS-Item returned by a previous call to get_wfs_catalog_item() .
---------------	---

Returns

the name corresponding to the WFS-Layer object

See also

[get_wfs_layer_title](#), [get_wfs_layer_abstract](#), [get_wfs_layer_srid_count](#), [get_wfs_layer_srid](#), [get_wfs_keyword_count](#), [get_wfs_keyword](#)

6.13.2.12 get_wfs_item_title()

```
SPATIALITE_DECLARE const char* get_wfs_item_title (  
    gaiaWFSitemPtr handle )
```

Return the title corresponding to some WFS-Item (aka Layer) object.

Parameters

<i>handle</i>	the pointer to a valid WFS-Item returned by a previous call to get_wfs_catalog_item() .
---------------	---

Returns

the title corresponding to the WFS-Layer object

See also

[get_wfs_item_name](#), [get_wfs_item_abstract](#), [get_wfs_layer_srid_count](#), [get_wfs_layer_srid](#), [get_wfs_keyword_count](#), [get_wfs_keyword](#)

6.13.2.13 get_wfs_keyword()

```
SPATIALITE_DECLARE const char* get_wfs_keyword (
    gaiaWFSitemPtr handle,
    int index )
```

Return one of the Keywords supported by a WFS-Item object.

Parameters

<i>handle</i>	the pointer to a valid WFS-Item returned by a previous call to get_wfs_catalog_item() .
<i>index</i>	the relative index identifying the required Keyword (the first Keyword associated to a WFS-Item object has index ZERO).

Returns

the Keyword value: NULL if the required Keyword isn't defined.

See also

[get_wfs_item_name](#), [get_wfs_item_title](#), [get_wfs_item_abstract](#), [get_wfs_layer_srid_count](#), [get_wfs_layer_srid](#), [get_wfs_layer_keyword](#)

6.13.2.14 get_wfs_keyword_count()

```
SPATIALITE_DECLARE int get_wfs_keyword_count (
    gaiaWFSitemPtr handle )
```

Return the total count of Keywords associated to a WFS-Item object.

Parameters

<i>handle</i>	the pointer to a valid WFS-Item returned by a previous call to get_wfs_catalog_item() .
---------------	---

Returns

the total count of Keyword associated to a WFS-Item object: a negative number if the WFS-Item isn't valid

See also

[get_wfs_item_name](#), [get_wfs_item_title](#), [get_wfs_item_abstract](#), [get_wfs_layer_srid_count](#), [get_wfs_layer_srid](#), [get_wfs_layer_keyword](#)

6.13.2.15 get_wfs_layer_srid()

```
SPATIALITE_DECLARE int get_wfs_layer_srid (
    gaiaWFSitemPtr handle,
    int index )
```

Return one of the SRIDs supported by a WFS-Item object.

Parameters

<i>handle</i>	the pointer to a valid WFS-Item returned by a previous call to get_wfs_catalog_item() .
<i>index</i>	the relative index identifying the required SRID (the first SRID value supported by a WFS-Item object has index ZERO).

Returns

the SRID-value: a negative number if the required SRID-value isn't defined.

See also

[get_wfs_item_name](#), [get_wfs_item_title](#), [get_wfs_item_abstract](#), [get_wfs_layer_srid_count](#), [get_wfs_keyword_count](#), [get_wfs_keyword](#)

6.13.2.16 get_wfs_layer_srid_count()

```
SPATIALITE_DECLARE int get_wfs_layer_srid_count (
    gaiaWFSitemPtr handle )
```

Return the total count of SRIDs supported by a WFS-Item object.

Parameters

<i>handle</i>	the pointer to a valid WFS-Item returned by a previous call to get_wfs_catalog_item() .
---------------	---

Returns

the total count of SRIDs supported by a WFS-Item object: a negative number if the WFS-Item isn't valid

See also

[get_wfs_item_name](#), [get_wfs_item_title](#), [get_wfs_item_abstract](#), [get_wfs_layer_srid](#), [get_wfs_keyword_count](#), [get_wfs_keyword](#)

6.13.2.17 get_wfs_request_url()

```
SPATIALITE_DECLARE char* get_wfs_request_url (
    gaiaWFScatalogPtr handle,
    const char * name,
    const char * version,
    int srid,
    int max_features )
```

Return a GetFeature URL (GET)

Parameters

<i>handle</i>	the pointer to a valid WFS-Item returned by a previous call to get_wfs_catalog_item() .
<i>name</i>	the NAME uniquely identifying the required WFS layer.
<i>version</i>	could be "1.0.0" or "1.1.0"; if NULL or invalid "1.1.0" will be assumed.
<i>srid</i>	the preferred SRS to be used for WFS geometries; if negative or mismatching will be simply ignored.
<i>max_features</i>	the WFS MAXFEATURES argument; any negative or zero value will be ignored.

Returns

the GetFeature URL: NULL if any error is found.

See also

[get_wfs_base_request_url](#), [get_wfs_describe_url](#)

Note

you are responsible to destroy (before or after) any allocated memory returned by [get_wfs_request_url\(\)](#).

6.13.2.18 get_wfs_schema_column()

```
SPATIALITE_DECLARE gaiaWFScolumnPtr get_wfs_schema_column (
    gaiaWFSschemaPtr handle,
    int index )
```

Return the pointer to some specific Column defined within a WFS-Schema object.

Parameters

<i>handle</i>	the pointer to a valid WFS-Schema returned by a previous call to create_wfs_schema()
<i>index</i>	the relative index identifying the required WFS-Column (the first Item in the WFS-Schema object has index ZERO).

Returns

the pointer to the required WFS-Column object: NULL if the passed index isn't valid

See also

[create_wfs_schema](#), [get_wfs_schema_geometry_info](#), [get_wfs_schema_column_count](#), [get_wfs_schema_column_info](#)

6.13.2.19 get_wfs_schema_column_count()

```
SPATIALITE_DECLARE int get_wfs_schema_column_count (
    gaiaWFSschemaPtr handle )
```

Return the total count of items (aka Columns) defined within a WFS-Schema object.

Parameters

<i>handle</i>	the pointer to a valid WFS-Schema returned by a previous call to create_wfs_schema()
---------------	--

Returns

the total count of items (aka Columns) defined within a WFS-Schema object: a negative number if the WFS-Schema isn't valid

See also

[create_wfs_schema](#), [get_wfs_schema_geometry_info](#), [get_wfs_schema_column](#), [get_wfs_schema_column_info](#)

6.13.2.20 get_wfs_schema_column_info()

```
SPATIALITE_DECLARE int get_wfs_schema_column_info (
    gaiaWFScolumnPtr handle,
    const char ** name,
    int * type,
    int * nullable )
```

Return the infos describing some WFS-Column object.

Parameters

<i>handle</i>	the pointer to a valid WFS-Column returned by a previous call to get_wfs_schema_column() .
<i>name</i>	on completion will contain a pointer to the Column name
<i>type</i>	on completion will contain the datatype set for the Column; could be one of SQLITE_TEXT, SQLITE_INTEGER or SQLITE_FLOAT
<i>nullable</i>	on completion will contain a Boolean value; if TRUE the Column may contain NULL-values.

Returns

TRUE on success, FALSE if any error is encountered

See also

[get_wfs_schema_column](#), [get_wfs_schema_geometry_info](#)

6.13.2.21 [get_wfs_schema_geometry_info\(\)](#)

```
SPATIALITE_DECLARE int get_wfs_schema_geometry_info (
    gaiaWFSschemaPtr handle,
    const char ** name,
    int * type,
    int * srid,
    int * dims,
    int * nullable )
```

Return the infos describing some WFS-GeometryColumn object.

Parameters

<i>handle</i>	the pointer to a valid WFS-Schema returned by a previous call to create_wfs_schema() .
<i>name</i>	on completion will contain a pointer to the GeometryColumn name
<i>type</i>	on completion will contain the GeometryType set for the Column; could be one of GAIA_POINT, GAIA_LINestring, GAIA_POLYGON, GAIA_MULTIPPOINT, GAIA_MULTILINESTRING, GAIA_MULTIPOLYGON or GAIA_GEOMETRYCOLLECTION
<i>srid</i>	on completion will contain the SRID-value set for the GeometryColumn
<i>dims</i>	on completion will contain the dimensions (2 or 3) set for the GeometryColumn
<i>nullable</i>	on completion will contain a Boolean value; if TRUE the Column may contain NULL-values.

Returns

TRUE on success, FALSE if any error is encountered or if the WFS-Schema hasn't any Geometry-Column defined.

See also

[create_wfs_schema](#), [get_wfs_schema_column_count](#), [get_wfs_schema_column](#), [get_wfs_schema_column_info](#)

6.13.2.22 `get_wfs_version()`

```
SPATIALITE_DECLARE const char* get_wfs_version (
    gaiaWFScatalogPtr handle )
```

Return the WFS-Version string as reported by GetCapabilities.

Parameters

<i>handle</i>	the pointer to a valid WFS-Item returned by a previous call to get_wfs_catalog_item() .
---------------	---

Returns

the WFS Version string: NULL is undefined

See also

[create_wfs_catalog](#)

6.13.2.23 `load_from_wfs()`

```
SPATIALITE_DECLARE int load_from_wfs (
    sqlite3 * sqlite,
    const char * path_or_url,
    const char * alt_describe_uri,
    const char * layer_name,
    int swap_axes,
    const char * table,
    const char * pk_column_name,
    int spatial_index,
    int * rows,
    char ** err_msg,
    void(*) (int, void *) progress_callback,
    void * callback_ptr )
```

Loads data from some WFS source.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>path_or_url</i>	pointer to some WFS-GetFeature XML Document (could be a pathname or an URL).
<i>alt_describe_uri</i>	an alternative URI for DescribeFeatureType to be used if no one is found within the XML document returned by GetFeature.
<i>layer_name</i>	the name of the WFS layer.
<i>swap_axes</i>	if TRUE the X and Y axes will be swapped
<i>table</i>	the name of the table to be created
<i>pk_column</i>	name of the Primary Key column; if NULL or mismatching then "PK_UID" will be assumed by default.
<i>spatial_index</i>	if TRUE an R*Tree Spatial Index will be created
<i>rows</i>	on completion will contain the total number of actually imported rows
<i>err_msg</i>	on completion will contain an error message (if any)
<i>progress_callback</i>	pointer to a callback function to be invoked immediately after processing each WFS page (could be NULL)
<i>callback_ptr</i>	an arbitrary pointer (to be passed as the second argument by the callback function).

See also

[create_wfs_catalog](#), [load_from_wfs_paged](#), [reset_wfs_http_connection](#)

Returns

0 on failure, any other value on success

Note

an eventual error message returned via `err_msg` requires to be deallocated by invoking `free()`
 please note: this one simply is a convenience method, and exactly corresponds to [load_from_wfs_paged\(\)](#) setting a negative page size.

6.13.2.24 load_from_wfs_paged()

```
SPATIALITE_DECLARE int load_from_wfs_paged (
    sqlite3 * sqlite,
    const char * path_or_url,
    const char * alt_describe_uri,
    const char * layer_name,
    int swap_axes,
    const char * table,
    const char * pk_column_name,
    int spatial_index,
    int page_size,
    int * rows,
    char ** err_msg,
    void(*) (int, void *) progress_callback,
    void * callback_ptr )
```

Loads data from some WFS source (using WFS paging)

Parameters

<i>sqlite</i>	handle to current DB connection
<i>path_or_url</i>	pointer to some WFS-GetFeature XML Document (could be a pathname or an URL).
<i>alt_describe_uri</i>	an alternative URI for DescribeFeatureType to be used if no one is found within the XML document returned by GetFeature.
<i>layer_name</i>	the name of the WFS layer.
<i>swap_axes</i>	if TRUE the X and Y axes will be swapped
<i>table</i>	the name of the table to be created
<i>pk_column</i>	name of the Primary Key column; if NULL or mismatching then "PK_UID" will be assumed by default.
<i>spatial_index</i>	if TRUE an R*Tree Spatial Index will be created
<i>page_size</i>	max number of features for each single WFS call; if zero or negative a single monolithic page is assumed (i.e. paging will not be applied).
<i>rows</i>	on completion will contain the total number of actually imported rows
<i>err_msg</i>	on completion will contain an error message (if any)
<i>progress_callback</i>	pointer to a callback function to be invoked immediately after processing each WFS page (could be NULL)
<i>callback_ptr</i>	an arbitrary pointer (to be passed as the second argument by the callback function).

See also

[create_wfs_catalog](#), [load_from_wfs](#), [reset_wfs_http_connection](#), [load_from_wfs_paged_ex](#)

Returns

0 on failure, any other value on success

Note

an eventual error message returned via `err_msg` requires to be deallocated by invoking `free()`

the `progress_callback` function must have this signature: **`void myfunct(int count, void *ptr)`**; and will cyclically report how many features have been processed since the initial call start.

this function is now deprecated and simply defaults to a call to `load_from_wfs_paged_ex` assuming WFS version 1.1.0

6.13.2.25 load_from_wfs_paged_ex()

```
SPATIALITE_DECLARE int load_from_wfs_paged_ex (
    sqlite3 * sqlite,
    const char * wfs_version,
    const char * path_or_url,
    const char * alt_describe_uri,
    const char * layer_name,
    int swap_axes,
    const char * table,
    const char * pk_column_name,
    int spatial_index,
    int page_size,
    int * rows,
    char ** err_msg,
    void(*) (int, void *) progress_callback,
    void * callback_ptr )
```

Loads data from some WFS source (using WFS paging) - Extended.

Parameters

<i>sqlite</i>	handle to current DB connection
<i>wfs_version</i>	one of "1.0.0", "1.1.0", "2.0.0" or "2.0.2"
<i>path_or_url</i>	pointer to some WFS-GetFeature XML Document (could be a pathname or an URL).
<i>alt_describe_uri</i>	an alternative URI for DescribeFeatureType to be used if no one is found within the XML document returned by GetFeature.
<i>layer_name</i>	the name of the WFS layer.
<i>swap_axes</i>	if TRUE the X and Y axes will be swapped
<i>table</i>	the name of the table to be created
<i>pk_column</i>	name of the Primary Key column; if NULL or mismatching then "PK_UID" will be assumed by default.
<i>spatial_index</i>	if TRUE an R*Tree Spatial Index will be created
<i>page_size</i>	max number of features for each single WFS call; if zero or negative a single monolithic page is assumed (i.e. paging will not be applied).
<i>rows</i>	on completion will contain the total number of actually imported rows
<i>err_msg</i>	on completion will contain an error message (if any)
<i>progress_callback</i>	pointer to a callback function to be invoked immediately after processing each WFS page (could be NULL)
<i>callback_ptr</i>	pointer to a callback function to be invoked immediately after processing each WFS page (could be NULL)

See also

[create_wfs_catalog](#), [load_from_wfs](#), [reset_wfs_http_connection](#)

Returns

0 on failure, any other value on success

Note

an eventual error message returned via `err_msg` requires to be deallocated by invoking `free()`
the `progress_callback` function must have this signature: **`void myfunct(int count, void *ptr)`**;
and will cyclically report how many features have been processed since the initial call start.

6.13.2.26 reset_wfs_http_connection()

```
SPATIALITE_DECLARE void reset_wfs_http_connection (  
    void )
```

Resets the libxml2 "nano HTTP": useful when changing the HTTP_PROXY settings.

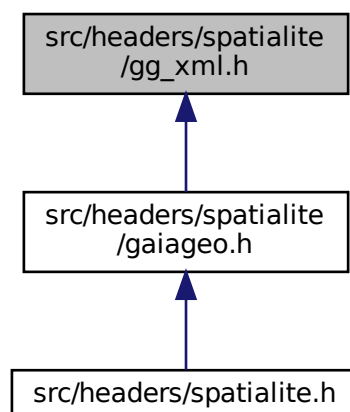
See also

[create_wfs_catalog](#), [load_from_wfs](#), [load_from_wfs_paged](#)

6.14 src/headers/spatialite/gg_xml.h File Reference

Geometry handling functions: XML document.

This graph shows which files directly or indirectly include this file:



Macros

- #define [GAIA_XML_START](#) 0x00
XmlBLOB internal marker: START.
- #define [GAIA_XML_END](#) 0xDD
XmlBLOB internal marker: END.
- #define [GAIA_XML_HEADER](#) 0xAC
XmlBLOB internal marker: HEADER.
- #define [GAIA_XML_LEGACY_HEADER](#) 0xAB
XmlBLOB internal marker: LEGACY HEADER.
- #define [GAIA_XML_SCHEMA](#) 0xBA
XmlBLOB internal marker: SCHEMA.
- #define [GAIA_XML_FILEID](#) 0xCA
XmlBLOB internal marker: FILEID.
- #define [GAIA_XML_PARENTID](#) 0xDA
XmlBLOB internal marker: PARENTID.
- #define [GAIA_XML_NAME](#) 0xDE
XmlBLOB internal marker: TITLE.
- #define [GAIA_XML_TITLE](#) 0xDB
XmlBLOB internal marker: TITLE.
- #define [GAIA_XML_ABSTRACT](#) 0xDC
XmlBLOB internal marker: ABSTRACT.
- #define [GAIA_XML_GEOMETRY](#) 0xDD
XmlBLOB internal marker: GEOMETRY.
- #define [GAIA_XML_CRC32](#) 0xBC
XmlBLOB internal marker: CRC32.
- #define [GAIA_XML_PAYLOAD](#) 0xCB
XmlBLOB internal marker: PAYLOAD.
- #define [GAIA_XML_LITTLE_ENDIAN](#) 0x01
XmlBLOB FLAG - LITTLE_ENDIAN bitmask.
- #define [GAIA_XML_COMPRESSED](#) 0x02
XmlBLOB FLAG - COMPRESSED bitmask.
- #define [GAIA_XML_VALIDATED](#) 0x04
XmlBLOB FLAG - VALIDATED bitmask.
- #define [GAIA_XML_ISO_METADATA](#) 0x80
XmlBLOB FLAG - ISO METADATA bitmask.
- #define [GAIA_XML_SLD_SE_RASTER_STYLE](#) 0x10
XmlBLOB FLAG - SLDSE VECTOR STYLE bitmask.
- #define [GAIA_XML_SLD_SE_VECTOR_STYLE](#) 0x40
XmlBLOB FLAG - SLDSE VECTOR STYLE bitmask.
- #define [GAIA_XML_SLD_STYLE](#) 0x48
XmlBLOB FLAG - SLD STYLE bitmask.
- #define [GAIA_XML_SVG](#) 0x20
XmlBLOB FLAG - SVG bitmask.
- #define [GAIA_XML_GPX](#) 0x08
XmlBLOB FLAG - GPX bitmask.
- #define [GAIA_XML_MAP_CONFIG](#) 0x88
XmlBLOB FLAG - MapConfig bitmask.

Functions

- GAIAGEO_DECLARE char * [gaia_libxml2_version](#) (void)
return the LIBXML2 version string
- GAIAGEO_DECLARE void [gaiaXmlToBlob](#) (const void *p_cache, const unsigned char *xml, int xml_len, int compressed, const char *schemaURI, unsigned char **result, int *size, char **parsing_errors, char **schema_validation_errors)
Creates an XmlBLOB buffer.
- GAIAGEO_DECLARE char * [gaiaXmlTextFromBlob](#) (const unsigned char *blob, int size, int indent)
Extract an XmlDocument from within an XmlBLOB buffer.
- GAIAGEO_DECLARE void [gaiaXmlFromBlob](#) (const unsigned char *blob, int size, int indent, unsigned char **result, int *res_size)
Extract an XmlDocument from within an XmlBLOB buffer.
- GAIAGEO_DECLARE int [gaialsValidXmlBlob](#) (const unsigned char *blob, int size)
Checks if a BLOB actually is a valid XmlBLOB buffer.
- GAIAGEO_DECLARE int [gaialsCompressedXmlBlob](#) (const unsigned char *blob, int size)
Checks if a valid XmlBLOB buffer is compressed or not.
- GAIAGEO_DECLARE int [gaialsIsoMetadataXmlBlob](#) (const unsigned char *blob, int size)
Checks if a valid XmlBLOB buffer does contain an ISO Metadata or not.
- GAIAGEO_DECLARE int [gaialsSldSeVectorStyleXmlBlob](#) (const unsigned char *blob, int size)
Checks if a valid XmlBLOB buffer does contain an SLD/SE Style or not.
- GAIAGEO_DECLARE int [gaialsSldSeRasterStyleXmlBlob](#) (const unsigned char *blob, int size)
Checks if a valid XmlBLOB buffer does contain an SLD/SE Style or not.
- GAIAGEO_DECLARE int [gaialsSldStyleXmlBlob](#) (const unsigned char *blob, int size)
Checks if a valid XmlBLOB buffer does contain an SLD Style or not.
- GAIAGEO_DECLARE int [gaialsMapConfigXmlBlob](#) (const unsigned char *blob, int size)
Checks if a valid XmlBLOB buffer does contain a MapConfig or not.
- GAIAGEO_DECLARE int [gaialsSvgXmlBlob](#) (const unsigned char *blob, int size)
Checks if a valid XmlBLOB buffer does contain an SVG Symbol or not.
- GAIAGEO_DECLARE int [gaialsGpxXmlBlob](#) (const unsigned char *blob, int size)
Checks if a valid XmlBLOB buffer does contain a GPX document or not.
- GAIAGEO_DECLARE void [gaiaXmlBlobCompression](#) (const unsigned char *blob, int in_size, int compressed, unsigned char **result, int *out_size)
Return another XmlBLOB buffer compressed / uncompressed.
- GAIAGEO_DECLARE int [gaialsSchemaValidatedXmlBlob](#) (const unsigned char *blob, int size)
Checks if a valid XmlBLOB buffer has successfully passed a formal Schema validation or not.
- GAIAGEO_DECLARE int [gaiaXmlBlobGetDocumentSize](#) (const unsigned char *blob, int size)
Return the XmlDocument size (in bytes) from a valid XmlBLOB buffer.
- GAIAGEO_DECLARE char * [gaiaXmlBlobGetSchemaURI](#) (const unsigned char *blob, int size)
Return the SchemaURI from a valid XmlBLOB buffer.
- GAIAGEO_DECLARE char * [gaiaXmlGetInternalSchemaURI](#) (const void *p_cache, const unsigned char *xml, int xml_len)
Return the Internal SchemaURI from a valid XmlDocument.
- GAIAGEO_DECLARE char * [gaiaXmlBlobGetFileId](#) (const unsigned char *blob, int size)
Return the FileIdentifier from a valid XmlBLOB buffer.
- GAIAGEO_DECLARE char * [gaiaXmlBlobGetParentId](#) (const unsigned char *blob, int size)
Return the ParentIdentifier from a valid XmlBLOB buffer.
- GAIAGEO_DECLARE int [gaiaXmlBlobSetFileId](#) (const void *p_cache, const unsigned char *blob, int size, const char *identifier, unsigned char **new_blob, int *new_size)
Return a new XmlBLOB (ISO Metadata) by replacing the FileId value.
- GAIAGEO_DECLARE int [gaiaXmlBlobSetParentId](#) (const void *p_cache, const unsigned char *blob, int size, const char *identifier, unsigned char **new_blob, int *new_size)

Return a new XmlBLOB (ISO Metadata) by replacing the ParentId value.

- GAIAGEO_DECLARE int [gaiaXmlBlobAddFileId](#) (const void *p_cache, const unsigned char *blob, int size, const char *identifier, const char *ns_id, const char *uri_id, const char *ns_charstr, const char *uri_charstr, unsigned char **new_blob, int *new_size)

Return a new XmlBLOB (ISO Metadata) by inserting a FileId value.

- GAIAGEO_DECLARE int [gaiaXmlBlobAddParentId](#) (const void *p_cache, const unsigned char *blob, int size, const char *identifier, const char *ns_id, const char *uri_id, const char *ns_charstr, const char *uri_charstr, unsigned char **new_blob, int *new_size)

Return a new XmlBLOB (ISO Metadata) by inserting a ParentId value.

- GAIAGEO_DECLARE char * [gaiaXmlBlobGetName](#) (const unsigned char *blob, int size)

Return the Name from a valid XmlBLOB buffer.

- GAIAGEO_DECLARE char * [gaiaXmlBlobGetTitle](#) (const unsigned char *blob, int size)

Return the Title from a valid XmlBLOB buffer.

- GAIAGEO_DECLARE char * [gaiaXmlBlobGetAbstract](#) (const unsigned char *blob, int size)

Return the Abstract from a valid XmlBLOB buffer.

- GAIAGEO_DECLARE void [gaiaXmlBlobGetGeometry](#) (const unsigned char *blob, int size, unsigned char **blob_geom, int *blob_size)

Return the Geometry buffer from a valid XmlBLOB buffer.

- GAIAGEO_DECLARE [gaiaGeomCollPtr](#) [gaiaXmlBlobMLineFromGPX](#) (const unsigned char *blob, int size, sqlite3 *db_handle)

Return a MultiLinestring Geometry from a valid GPX XmlBLOB buffer.

- GAIAGEO_DECLARE char * [gaiaXmlBlobGetEncoding](#) (const unsigned char *blob, int size)

Return the Charset Encoding from a valid XmlBLOB buffer.

- GAIAGEO_DECLARE char * [gaiaXmlBlobGetLastParseError](#) (const void *p_cache)

Return the most recent XML Parse error/warning (if any)

- GAIAGEO_DECLARE char * [gaiaXmlBlobGetLastValidateError](#) (const void *p_cache)

Return the most recent XML Validate error/warning (if any)

- GAIAGEO_DECLARE int [gaialsValidXPathExpression](#) (const void *p_cache, const char *xpath_expr)

Checks if a Text string could be a valid XPathExpression.

- GAIAGEO_DECLARE char * [gaiaXmlBlobGetLastXPathError](#) (const void *p_cache)

Return the most recent XPath error/warning (if any)

- GAIAGEO_DECLARE int [gaiaXmlLoad](#) (const void *p_cache, const char *path_or_url, unsigned char **result, int *size, char **parsing_errors)

Load an external XML Document.

- GAIAGEO_DECLARE int [gaiaXmlStore](#) (const unsigned char *blob, int size, const char *path, int indent)

Stores an external XML Document.

6.14.1 Detailed Description

Geometry handling functions: XML document.

6.14.2 Function Documentation

6.14.2.1 `gaia_libxml2_version()`

```
GAIA GEO_DECLARE char* gaia_libxml2_version (
    void )
```

return the LIBXML2 version string

Returns

a text string identifying the current LIBXML2 version

Note

the version string corresponds to dynamically allocated memory: so you are responsible to `free()` it [unless SQLite will take care of memory cleanup via buffer binding].

6.14.2.2 `gaialsCompressedXmlBlob()`

```
GAIA GEO_DECLARE int gaiaIsCompressedXmlBlob (
    const unsigned char * blob,
    int size )
```

Checks if a valid XmlBLOB buffer is compressed or not.

Parameters

<i>blob</i>	pointer to the XmlBLOB buffer.
<i>size</i>	XmlBLOB's size (in bytes).

Returns

TRUE or FALSE if the BLOB actually is a valid XmlBLOB; -1 in any other case.

See also

[gaialsValidXmlBlob](#), [gaialsSchemaValidatedXmlBlob](#), [gaialsIsoMetadataXmlBlob](#), [gaialsSldSeVectorStyleXmlBlob](#), [gaialsSldSeRasterStyleXmlBlob](#), [gaialsSldStyleXmlBlob](#), [gaialsSvgXmlBlob](#), [gaialsGpxXmlBlob](#), [gaialsMapConfigXmlBlob](#)

6.14.2.3 `gaialsGpxXmlBlob()`

```
GAIA GEO_DECLARE int gaiaIsGpxXmlBlob (
    const unsigned char * blob,
    int size )
```

Checks if a valid XmlBLOB buffer does contain a GPX document or not.

Parameters

<i>blob</i>	pointer to the XmlBLOB buffer.
<i>size</i>	XmlBLOB's size (in bytes).

Returns

TRUE or FALSE if the BLOB actually is a valid XmlBLOB; -1 in any other case.

See also

[gaiaIsValidXmlBlob](#), [gaiaIsSchemaValidatedXmlBlob](#), [gaiaIsCompressedXmlBlob](#), [gaiaIsIsoMetadataXmlBlob](#), [gaiaIsSldSeVectorStyleXmlBlob](#), [gaiaIsSldStyleXmlBlob](#), [gaiaIsSldSeRasterStyleXmlBlob](#), [gaiaIsSvgXmlBlob](#), [gaiaIsMapConfigXmlBlob](#)

6.14.2.4 gaiaIsIsoMetadataXmlBlob()

```
GAIA_GEO_DECLARE int gaiaIsIsoMetadataXmlBlob (
    const unsigned char * blob,
    int size )
```

Checks if a valid XmlBLOB buffer does contain an ISO Metadata or not.

Parameters

<i>blob</i>	pointer to the XmlBLOB buffer.
<i>size</i>	XmlBLOB's size (in bytes).

Returns

TRUE or FALSE if the BLOB actually is a valid XmlBLOB; -1 in any other case.

See also

[gaiaIsValidXmlBlob](#), [gaiaIsSchemaValidatedXmlBlob](#), [gaiaIsCompressedXmlBlob](#), [gaiaIsSldSeVectorStyleXmlBlob](#), [gaiaIsSldSeRasterStyleXmlBlob](#), [gaiaIsSldStyleXmlBlob](#), [gaiaIsSvgXmlBlob](#), [gaiaIsGpxXmlBlob](#), [gaiaIsMapConfigXmlBlob](#)

6.14.2.5 gaiaIsMapConfigXmlBlob()

```
GAIA_GEO_DECLARE int gaiaIsMapConfigXmlBlob (
    const unsigned char * blob,
    int size )
```

Checks if a valid XmlBLOB buffer does contain a MapConfig or not.

Parameters

<i>blob</i>	pointer to the XmlBLOB buffer.
<i>size</i>	XmlBLOB's size (in bytes).

Returns

TRUE or FALSE if the BLOB actually is a valid XmlBLOB of the MapConfig type; -1 in any other case.

See also

[gaialsValidXmlBlob](#), [gaialsSchemaValidatedXmlBlob](#), [gaialsCompressedXmlBlob](#), [gaialsIsoMetadataXmlBlob](#), [gaialsSldSeVectorStyleXmlBlob](#), [gaialsSldSeRasterXmlBlob](#), [gaialsSvgXmlBlob](#), [gaialsGpxXmlBlob](#), [gaialsSldStyleXmlBlob](#)

6.14.2.6 gaialsSchemaValidatedXmlBlob()

```
GAIA GEO_DECLARE int gaiaIsSchemaValidatedXmlBlob (
    const unsigned char * blob,
    int size )
```

Checks if a valid XmlBLOB buffer has successfully passed a formal Schema validation or not.

Parameters

<i>blob</i>	pointer to the XmlBLOB buffer.
<i>size</i>	XmlBLOB's size (in bytes).

Returns

TRUE or FALSE if the BLOB actually is a valid XmlBLOB but not schema-validated; -1 in any other case.

See also

[gaialsValidXmlBlob](#), [gaialsSvgXmlBlob](#), [gaialsCompressedXmlBlob](#), [gaialsIsoMetadataXmlBlob](#), [gaialsSldSeVectorStyleXmlBlob](#), [gaialsSldSeRasterStyleXmlBlob](#), [gaialsSldStyleXmlBlob](#), [gaialsMapConfigXmlBlob](#)

6.14.2.7 gaialsSldSeRasterStyleXmlBlob()

```
GAIA GEO_DECLARE int gaiaIsSldSeRasterStyleXmlBlob (
    const unsigned char * blob,
    int size )
```

Checks if a valid XmlBLOB buffer does contain an SLD/SE Style or not.

Parameters

<i>blob</i>	pointer to the XmlBLOB buffer.
<i>size</i>	XmlBLOB's size (in bytes).

Returns

TRUE or FALSE if the BLOB actually is a valid XmlBLOB of the Raster type; -1 in any other case.

See also

[gaialsValidXmlBlob](#), [gaialsSchemaValidatedXmlBlob](#), [gaialsCompressedXmlBlob](#), [gaialsIsoMetadataXmlBlob](#), [gaialsSldSeVectorStyleXmlBlob](#), [gaialsSldStyleXmlBlob](#), [gaialsSvgXmlBlob](#), [gaialsGpxXmlBlob](#), [gaialsMapConfigXmlBlob](#)

6.14.2.8 gaialsSldSeVectorStyleXmlBlob()

```
GAIA_GEO_DECLARE int gaiaIsSldSeVectorStyleXmlBlob (  
    const unsigned char * blob,  
    int size )
```

Checks if a valid XmlBLOB buffer does contain an SLD/SE Style or not.

Parameters

<i>blob</i>	pointer to the XmlBLOB buffer.
<i>size</i>	XmlBLOB's size (in bytes).

Returns

TRUE or FALSE if the BLOB actually is a valid XmlBLOB of the Vector type; -1 in any other case.

See also

[gaialsValidXmlBlob](#), [gaialsSchemaValidatedXmlBlob](#), [gaialsCompressedXmlBlob](#), [gaialsIsoMetadataXmlBlob](#), [gaialsSldSeRasterStyleXmlBlob](#), [gaialsSvgXmlBlob](#), [gaialsGpxXmlBlob](#)

6.14.2.9 gaialsSldStyleXmlBlob()

```
GAIA_GEO_DECLARE int gaiaIsSldStyleXmlBlob (  
    const unsigned char * blob,  
    int size )
```

Checks if a valid XmlBLOB buffer does contain an SLD Style or not.

Parameters

<i>blob</i>	pointer to the XmlBLOB buffer.
<i>size</i>	XmlBLOB's size (in bytes).

Returns

TRUE or FALSE if the BLOB actually is a valid XmlBLOB of the SLD type; -1 in any other case.

See also

[gaialsValidXmlBlob](#), [gaialsSchemaValidatedXmlBlob](#), [gaialsCompressedXmlBlob](#), [gaialsIsoMetadataXmlBlob](#), [gaialsSldSeVectorStyleXmlBlob](#), [gaialsSldSeRasterXmlBlob](#), [gaialsSvgXmlBlob](#), [gaialsGpxXmlBlob](#), [gaialsMapConfigXmlBlob](#)

6.14.2.10 gaialsSvgXmlBlob()

```
GAIA GEO_DECLARE int gaiaIsSvgXmlBlob (
    const unsigned char * blob,
    int size )
```

Checks if a valid XmlBLOB buffer does contain an SVG Symbol or not.

Parameters

<i>blob</i>	pointer to the XmlBLOB buffer.
<i>size</i>	XmlBLOB's size (in bytes).

Returns

TRUE or FALSE if the BLOB actually is a valid XmlBLOB; -1 in any other case.

See also

[gaialsValidXmlBlob](#), [gaialsSchemaValidatedXmlBlob](#), [gaialsCompressedXmlBlob](#), [gaialsIsoMetadataXmlBlob](#), [gaialsSldSeVectorStyleXmlBlob](#), [gaialsSldStyleXmlBlob](#), [gaialsSldSeRasterStyleXmlBlob](#), [gaialsGpxXmlBlob](#), [gaialsMapConfigXmlBlob](#)

6.14.2.11 gaialsValidXmlBlob()

```
GAIA GEO_DECLARE int gaiaIsValidXmlBlob (
    const unsigned char * blob,
    int size )
```

Checks if a BLOB actually is a valid XmlBLOB buffer.

Parameters

<i>blob</i>	pointer to the XmlBLOB buffer.
<i>size</i>	XmlBLOB's size (in bytes).

Returns

TRUE or FALSE

See also

[gaialsCompressedXmlBlob](#), [gaialsSchemaValidatedXmlBlob](#), [gaialsIsoMetadataXmlBlob](#), [gaialsSldSeVectorStyleXmlBlob](#), [gaialsSldSeRasterStyleXmlBlob](#), [gaialsSldStyleXmlBlob](#), [gaialsSvgXmlBlob](#), [gaialsGpxXmlBlob](#), [gaialsMapConfigXmlBlob](#)

6.14.2.12 `gaialsValidXPathExpression()`

```
GAIA GEO_DECLARE int gaiaIsValidXPathExpression (
    const void * p_cache,
    const char * xpath_expr )
```

Checks if a Text string could be a valid XPathExpression.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>xpath_expr</i>	pointer to the XPathExpression to be checked.

Returns

TRUE or FALSE if the Text string actually is a valid XPathExpression; -1 in any other case.

See also

[gaiaXmlBlobGetLastXPathError](#)

6.14.2.13 `gaiaXmlBlobAddFileId()`

```
GAIA GEO_DECLARE int gaiaXmlBlobAddFileId (
    const void * p_cache,
    const unsigned char * blob,
    int size,
    const char * identifier,
    const char * ns_id,
    const char * uri_id,
```

```

const char * ns_charstr,
const char * uri_charstr,
unsigned char ** new_blob,
int * new_size )

```

Return a new XmlBLOB (ISO Metadata) by inserting a FileId value.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>blob</i>	pointer to the input XmlBLOB buffer.
<i>size</i>	input XmlBLOB's size (in bytes).
<i>identifier</i>	the new FileId value to be inserted.
<i>ns_id</i>	prefix corresponding to FileIdentifier NameSpace (may be NULL)
<i>uri_id</i>	URI corresponding to the FileIdentifier NameSpace (may be NULL)
<i>ns_charstr</i>	prefix corresponding to CharacterString NameSpace (may be NULL)
<i>uri_charstr</i>	URI corresponding to CharacterString NameSpace (may be NULL)
<i>new_blob</i>	on completion will contain a pointer to the output XmlBLOB buffer.
<i>new_size</i>	on completion will contain the output XmlBlob's size (in bytes).

Returns

TRUE for success; FALSE for any failure cause.

See also

[gaiaIsoMetadataXmlBlob](#), [gaiaXmlBlobGetFileId](#), [gaiaXmlBlobSetFileId](#)

Note

the output XmlBLOB corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

6.14.2.14 gaiaXmlBlobAddParentId()

```

GAIAGEO_DECLARE int gaiaXmlBlobAddParentId (
    const void * p_cache,
    const unsigned char * blob,
    int size,
    const char * identifier,
    const char * ns_id,
    const char * uri_id,
    const char * ns_charstr,
    const char * uri_charstr,
    unsigned char ** new_blob,
    int * new_size )

```

Return a new XmlBLOB (ISO Metadata) by inserting a ParentId value.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>blob</i>	pointer to the inputXmlBLOB buffer.
<i>size</i>	input XmlBLOB's size (in bytes).
<i>identifier</i>	the new ParentId value to be inserted.
<i>ns_id</i>	prefix corresponding to FileIdentifier NameSpace (may be NULL)
<i>uri_id</i>	URI corresponding to the FileIdentifier NameSpace (may be NULL)
<i>ns_charstr</i>	prefix corresponding to CharacterString NameSpace (may be NULL)
<i>uri_charstr</i>	URI corresponding to CharacterString NameSpace (may be NULL)
<i>new_blob</i>	on completion will contain a pointer to the output XmlBLOB buffer.
<i>new_size</i>	on completion will contain the output XmlBlob's size (in bytes).

Returns

TRUE for success; FALSE for any failure cause.

See also

[gaiaIsoMetadataXmlBlob](#), [gaiaXmlBlobGetParentId](#), [gaiaXmlBlobSetParentId](#)

Note

the returned XmlBLOB corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

6.14.2.15 **gaiaXmlBlobCompression()**

```
GAIA GEO_DECLARE void gaiaXmlBlobCompression (
    const unsigned char * blob,
    int in_size,
    int compressed,
    unsigned char ** result,
    int * out_size )
```

Return another XmlBLOB buffer compressed / uncompressed.

Parameters

<i>blob</i>	pointer to the input XmlBLOB buffer.
<i>in_size</i>	input XmlBLOB's size (in bytes).
<i>compressed</i>	if TRUE the returned XmlBLOB will be zip-compressed.
<i>result</i>	on completion will contain a pointer to the output XmlBLOB: NULL on failure.
<i>out_size</i>	on completion this variable will contain the output XmlBLOB's size (in bytes)

See also

[gaiaXmlToBlob](#), [gaiaIsCompressedXmlBlob](#)

Note

the XmlBLOB buffer corresponds to dynamically allocated memory: so you are responsible to free() it [unless SQLite will take care of memory cleanup via buffer binding].

6.14.2.16 gaiaXmlBlobGetAbstract()

```
GAIA GEO_DECLARE char* gaiaXmlBlobGetAbstract (
    const unsigned char * blob,
    int size )
```

Return the Abstract from a valid XmlBLOB buffer.

Parameters

<i>blob</i>	pointer to the XmlBLOB buffer.
<i>size</i>	XmlBLOB's size (in bytes).

Returns

the Abstract for any valid XmlBLOB containing an Abstract; NULL in any other case.

See also

[gaiaIsIsoMetadataXmlBlob](#), [gaiaIsSldSeVectorStyleXmlBlob](#), [gaiaIsSldSeRasterStyleXmlBlob](#), [gaiaIsSldStyleXmlBlob](#)

Note

the returned Abstract corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

6.14.2.17 gaiaXmlBlobGetDocumentSize()

```
GAIA GEO_DECLARE int gaiaXmlBlobGetDocumentSize (
    const unsigned char * blob,
    int size )
```

Return the XMLDocument size (in bytes) from a valid XmlBLOB buffer.

Parameters

<i>blob</i>	pointer to the XmlBLOB buffer.
<i>size</i>	XmlBLOB's size (in bytes).

Returns

the XmlDocument size (in bytes) for any valid XmlBLOB; -1 if the BLOB isn't a valid XmlBLOB.

6.14.2.18 gaiaXmlBlobGetEncoding()

```
GAIA_GEO_DECLARE char* gaiaXmlBlobGetEncoding (  
    const unsigned char * blob,  
    int size )
```

Return the Charset Encoding from a valid XmlBLOB buffer.

Parameters

<i>blob</i>	pointer to the XmlBLOB buffer.
<i>size</i>	XmlBLOB's size (in bytes).

Returns

the Charset Encoding for any valid XmlBLOB explicitly defining an Encoding; NULL in any other case.

Note

the returned Encoding corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

6.14.2.19 gaiaXmlBlobGetFileId()

```
GAIA_GEO_DECLARE char* gaiaXmlBlobGetFileId (  
    const unsigned char * blob,  
    int size )
```

Return the FileIdentifier from a valid XmlBLOB buffer.

Parameters

<i>blob</i>	pointer to the XmlBLOB buffer.
<i>size</i>	XmlBLOB's size (in bytes).

Returns

the FileIdentifier for any valid XmlBLOB containing a FileIdentifier; NULL in any other case.

See also

[gaiaIsoMetadataXmlBlob](#), [gaiaXmlBlobSetFileId](#), [gaiaXmlBlobAddFileId](#)

Note

the returned FileIdentifier corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

6.14.2.20 gaiaXmlBlobGetGeometry()

```
GAIAGEO_DECLARE void gaiaXmlBlobGetGeometry (
    const unsigned char * blob,
    int size,
    unsigned char ** blob_geom,
    int * blob_size )
```

Return the Geometry buffer from a valid XmlBLOB buffer.

Parameters

<i>blob</i>	pointer to the XmlBLOB buffer.
<i>size</i>	XmlBLOB's size (in bytes).
<i>blob_geom</i>	on completion this variable will contain a pointer to the returned Geometry buffer (NULL if no Geometry was defined within the XmlBLOB)
<i>blob_size</i>	on completion this variable will contain the size (in bytes) of the returned Geometry buffer

See also

[gaiaIsoMetadataXmlBlob](#)

Note

the returned Geometry buffer corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

6.14.2.21 gaiaXmlBlobGetLastError()

```
GAIAGEO_DECLARE char* gaiaXmlBlobGetLastError (
    const void * p_cache )
```

Return the most recent XML Parse error/warning (if any)

Parameters

<i>ptr</i>	a memory pointer returned by spatialite_alloc_connection()
------------	--

Returns

the most recent XML Parse error/warning message (if any); NULL in any other case.

See also

[gaiaXmlBlobGetLastValidateError](#), [gaialsValidXPathExpression](#), [gaiaXmlBlobGetLastXPathError](#)

Note

the returned error/warning message corresponds to dynamically allocated memory: so you are responsible to `free()` it before or after.

6.14.2.22 `gaiaXmlBlobGetLastValidateError()`

```
GAIAGEO_DECLARE char* gaiaXmlBlobGetLastValidateError (
    const void * p_cache )
```

Return the most recent XML Validate error/warning (if any)

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
----------------	--

Returns

the most recent XML Validate error/warning message (if any); NULL in any other case.

See also

[gaiaXmlBlobGetLastParseError](#), [gaialsValidXPathExpression](#), [gaiaXmlBlobGetLastXPathError](#)

Note

the returned error/warning message corresponds to dynamically allocated memory: so you are responsible to `free()` it before or after.

6.14.2.23 `gaiaXmlBlobGetLastXPathError()`

```
GAIAGEO_DECLARE char* gaiaXmlBlobGetLastXPathError (
    const void * p_cache )
```

Return the most recent XPath error/warning (if any)

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
----------------	--

Returns

the most recent XPath error/warning message (if any); NULL in any other case.

See also

[gaiaXmlBlobGetLastError](#), [gaiaXmlBlobGetLastValidateError](#), [gaialsValidXPathExpression](#)

Note

the returned error/warning message corresponds to dynamically allocated memory: so you are responsible to `free()` it before or after.

6.14.2.24 gaiaXmlBlobGetName()

```
GAIAGEO_DECLARE char* gaiaXmlBlobGetName (
    const unsigned char * blob,
    int size )
```

Return the Name from a valid XmlBLOB buffer.

Parameters

<i>blob</i>	pointer to the XmlBLOB buffer.
<i>size</i>	XmlBLOB's size (in bytes).

Returns

the Name for any valid XmlBLOB containing a Name; NULL in any other case.

See also

[gaialsIsoMetadataXmlBlob](#), [gaialsSldSeVectorStyleXmlBlob](#), [gaialsSldSeRasterStyleXmlBlob](#), [gaialsSldStyleXmlBlob](#)

Note

the returned Name corresponds to dynamically allocated memory: so you are responsible to `free()` it before or after.

6.14.2.25 `gaiaXmlBlobGetParentId()`

```
GAIA GEO_DECLARE char* gaiaXmlBlobGetParentId (
    const unsigned char * blob,
    int size )
```

Return the ParentIdentifier from a valid XmlBLOB buffer.

Parameters

<i>blob</i>	pointer to the XmlBLOB buffer.
<i>size</i>	XmlBLOB's size (in bytes).

Returns

the ParentIdentifier for any valid XmlBLOB containing a ParentIdentifier; NULL in any other case.

See also

[gaiaIsoMetadataXmlBlob](#), [gaiaXmlBlobSetParentId](#), [gaiaXmlBlobAddParentId](#)

Note

the returned ParentIdentifier corresponds to dynamically allocated memory: so you are responsible to `free()` it before or after.

6.14.2.26 `gaiaXmlBlobGetSchemaURI()`

```
GAIA GEO_DECLARE char* gaiaXmlBlobGetSchemaURI (
    const unsigned char * blob,
    int size )
```

Return the SchemaURI from a valid XmlBLOB buffer.

Parameters

<i>blob</i>	pointer to the XmlBLOB buffer.
<i>size</i>	XmlBLOB's size (in bytes).

Returns

the SchemaURI for any valid XmlBLOB containing a SchemaURI; NULL in any other case.

See also

[gaiaXmlGetInternalSchemaURI](#)

Note

the returned SchemaURI corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

6.14.2.27 gaiaXmlBlobGetTitle()

```
GAIAGEO_DECLARE char* gaiaXmlBlobGetTitle (
    const unsigned char * blob,
    int size )
```

Return the Title from a valid XmlBLOB buffer.

Parameters

<i>blob</i>	pointer to the XmlBLOB buffer.
<i>size</i>	XmlBLOB's size (in bytes).

Returns

the Title for any valid XmlBLOB containing a Title; NULL in any other case.

See also

[gaiaIsoMetadataXmlBlob](#), [gaiaIsSldSeVectorStyleXmlBlob](#), [gaiaIsSldSeRasterStyleXmlBlob](#), [gaiaIsSldStyleXmlBlob](#)

Note

the returned Title corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

6.14.2.28 gaiaXmlBlobMLineFromGPX()

```
GAIAGEO_DECLARE gaiaGeomCollPtr gaiaXmlBlobMLineFromGPX (
    const unsigned char * blob,
    int size,
    sqlite3 * db_handle )
```

Return a MultiLinestring Geometry from a valid GPX XmlBLOB buffer.

Parameters

<i>blob</i>	pointer to the XmlBLOB buffer.
<i>size</i>	XmlBLOB's size (in bytes).
<i>db_handle</i>	handle to the current SQLite connection

Returns

a Geometry of the MultiLinestring type, or NULL

See also

[gaiaIsoMetadataXmlBlob](#)

Note

the returned Geometry corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

6.14.2.29 gaiaXmlBlobSetFileId()

```
GAIA GEO_DECLARE int gaiaXmlBlobSetFileId (
    const void * p_cache,
    const unsigned char * blob,
    int size,
    const char * identifier,
    unsigned char ** new_blob,
    int * new_size )
```

Return a new XmlBLOB (ISO Metadata) by replacing the FileId value.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>blob</i>	pointer to the input XmlBLOB buffer.
<i>size</i>	input XmlBLOB's size (in bytes).
<i>identifier</i>	the new FileId value to be set.
<i>new_blob</i>	on completion will contain a pointer to the output XmlBLOB buffer.
<i>new_size</i>	on completion will contain the output XmlBlob's size (in bytes).

Returns

TRUE for success; FALSE for any failure cause.

See also

[gaiaIsoMetadataXmlBlob](#), [gaiaXmlBlobGetFileId](#), [gaiaXmlBlobAddFileId](#)

Note

the output XmlBLOB corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

6.14.2.30 `gaiaXmlBlobSetParentId()`

```
GAIA GEO_DECLARE int gaiaXmlBlobSetParentId (
    const void * p_cache,
    const unsigned char * blob,
    int size,
    const char * identifier,
    unsigned char ** new_blob,
    int * new_size )
```

Return a new XmlBLOB (ISO Metadata) by replacing the ParentId value.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>blob</i>	pointer to the input XmlBLOB buffer.
<i>size</i>	input XmlBLOB's size (in bytes).
<i>identifier</i>	the new ParentId value to be set.
<i>new_blob</i>	on completion will contain a pointer to the output XmlBLOB buffer.
<i>new_size</i>	on completion will contain the output XmlBlob's size (in bytes).

Returns

TRUE for success; FALSE for any failure cause.

See also

[gaiaIsoMetadataXmlBlob](#), [gaiaXmlBlobGetParentId](#), [gaiaXmlBlobAddParentId](#)

Note

the returned XmlBLOB corresponds to dynamically allocated memory: so you are responsible to `free()` it before or after.

6.14.2.31 `gaiaXmlFromBlob()`

```
GAIA GEO_DECLARE void gaiaXmlFromBlob (
    const unsigned char * blob,
    int size,
    int indent,
    unsigned char ** result,
    int * res_size )
```

Extract an XMLDocument from within an XmlBLOB buffer.

Parameters

<i>blob</i>	pointer to the XmlBLOB buffer.
<i>size</i>	XmlBLOB's size (in bytes).
<i>indent</i>	if a negative value is passed the XMLDocument will be extracted exactly as it was when loaded. Otherwise it will be properly formatted using the required indenting (max. 8); ZERO means that the whole XML Document will consist of a single line.
<i>result</i>	pointer to the memory buffer containing the XML Document
<i>res_size</i>	dimension (in bytes) of the XML Document memory buffer (both values will be passed back after

See also

[gaiaXmlToBlob](#), [gaiaXmlTextFromBlob](#)

Note

the returned XMLDocument will always respect the internal encoding declaration, and may not support any further processing as SQLite TEXT if it's not UTF-8.

the XMLDocument buffer corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

6.14.2.32 `gaiaXmlGetInternalSchemaURI()`

```
GAIA GEO_DECLARE char* gaiaXmlGetInternalSchemaURI (
    const void * p_cache,
    const unsigned char * xml,
    int xml_len )
```

Return the Internal SchemaURI from a valid XmlDocument.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>xml</i>	pointer to the XML document
<i>xml_len</i>	length of the XML document (in bytes).

Returns

the SchemaURI eventually defined within a valid XMLDocument; NULL if the XMLDocument is invalid, or if it doesn't contain any SchemaURI.

See also

[gaiaXmlBlobGetSchemaURI](#)

Note

the returned SchemaURI corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

6.14.2.33 `gaiaXmlLoad()`

```
GAIA GEO_DECLARE int gaiaXmlLoad (
    const void * p_cache,
    const char * path_or_url,
    unsigned char ** result,
    int * size,
    char ** parsing_errors )
```

Load an external XML Document.

Parameters

<i>path_or_url</i>	pointer to the external XML Document (could be a pathname or an URL).
<i>result</i>	on completion will contain a pointer to a BLOB: NULL on failure.
<i>size</i>	on completion this variable will contain the BLOB's size (in bytes).
<i>parsing_errors</i>	on completion this variable will contain all error/warning messages emitted during the XML Parsing step. Can be set to NULL so to ignore any message.

See also

[gaiaXmlFromBlob](#), [gaiaXmlStore](#)

Note

the BLOB buffer corresponds to dynamically allocated memory: so you are responsible to free() it [unless SQLite will take care of memory cleanup via buffer binding].

6.14.2.34 **gaiaXmlStore()**

```
GAIA GEO_DECLARE int gaiaXmlStore (
    const unsigned char * blob,
    int size,
    const char * path,
    int indent )
```

Stores an external XML Document.

Parameters

<i>blob</i>	pointer to the XmlBLOB buffer.
<i>size</i>	XmlBLOB's size (in bytes).
<i>path</i>	pathname of the export file
<i>indent</i>	if a negative value is passed the XMLDocument will be extracted exactly as it was when loaded. Otherwise it will be properly formatted using the required indenting (max. 8); ZERO means that the whole XML Document will consist of a single line.

See also

[gaiaXmlToBlob](#), [gaiaXmlTextFromBlob](#)

Note

the returned XMLDocument will always respect the internal encoding declaration, and may not support any further processing as SQLite TEXT if it's not UTF-8.

the XMLDocument buffer corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

See also

[gaiaXmlFromBlob](#), [gaiaXmlLoad](#)

Note

the BLOB buffer corresponds to dynamically allocated memory: so you are responsible to free() it [unless SQLite will take care of memory cleanup via buffer binding].

6.14.2.35 `gaiaXmlTextFromBlob()`

```
GAIA GEO_DECLARE char* gaiaXmlTextFromBlob (
    const unsigned char * blob,
    int size,
    int indent )
```

Extract an XMLDocument from within an XmlBLOB buffer.

Parameters

<i>blob</i>	pointer to the XmlBLOB buffer.
<i>size</i>	XmlBLOB's size (in bytes).
<i>indent</i>	if a negative value is passed the XMLDocument will be extracted exactly as it was when loaded. Otherwise it will be properly formatted using the required indenting (max. 8); ZERO means that the whole XML Document will consist of a single line.

Returns

the pointer to the newly created XMLDocument buffer: NULL on failure

See also

[gaiaXmlToBlob](#), [gaiaXmlFromBlob](#)

Note

the returned XMLDocument will always be encoded as UTF-8 (irrespectively from the internal encoding declaration), so to allow any further processing as SQLite TEXT.

the XMLDocument buffer corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

6.14.2.36 `gaiaXmlToBlob()`

```
GAIA GEO_DECLARE void gaiaXmlToBlob (
    const void * p_cache,
    const unsigned char * xml,
    int xml_len,
    int compressed,
    const char * schemaURI,
    unsigned char ** result,
    int * size,
    char ** parsing_errors,
    char ** schema_validation_errors )
```

Creates an XmlBLOB buffer.

Parameters

<i>p_cache</i>	a memory pointer returned by spatialite_alloc_connection()
<i>xml</i>	pointer to the XML document (XmlBLOB payload).
<i>xml_len</i>	length of the XML document (in bytes).
<i>compressed</i>	if TRUE the returned XmlBLOB will be zip-compressed.
<i>schemaURI</i>	if not NULL the XML document will be assumed to be valid only if it successfully passes a formal Schema validation.
<i>result</i>	on completion will contain a pointer to XmlBLOB: NULL on failure.
<i>size</i>	on completion this variable will contain the XmlBLOB's size (in bytes)
<i>parsing_errors</i>	on completion this variable will contain all error/warning messages emitted during the XML Parsing step. Can be set to NULL so to ignore any message.
<i>schema_validation_errors</i>	on completion this variable will contain all error/warning messages emitted during the XML Schema Validation step. Can be set to NULL so to ignore any message.

See also

[gaiaXmlFromBlob](#), [gaiaXmlTextFromBlob](#), [gaiaXmlBlobGetLastParseError](#), [gaiaXmlBlobGetLastValidateError](#)

Note

the XmlBLOB buffer corresponds to dynamically allocated memory: so you are responsible to `free()` it [unless SQLite will take care of memory cleanup via buffer binding].

Chapter 7

Example Documentation

7.1 demo1.c

This is a sample C source showing how to use SQLite / SpatiaLite from C. This program shows the basic functionality that will be required for most SpatiaLite programs:

- how to connect an SQLite+SpatiaLite database
- executing an SQL query
- fetching values from a result set
- transforming BLOB-values into GEOMETRY
- elementary processing GEOMETRY

The typical output of this demo is shown below, when run against the sample database.

```
$ ./demo1 test-2.3.sqlite
SQLite version: 3.7.4
SpatiaLite version: 3.0.0-beta1

===== table 'HighWays' =====
row #1
    PK_UID      = 1
    Name        = 'Unknown'
    Geometry     = LINESTRING SRID=32632 length=8697.57
row #2
    PK_UID      = 2
    Name        = 'Unknown'
    Geometry     = LINESTRING SRID=32632 length=39.79
row #3
    PK_UID      = 3
    Name        = 'Unknown'
    Geometry     = LINESTRING SRID=32632 length=14610.39
row #4
    PK_UID      = 4
    Name        = 'Unknown'
    Geometry     = LINESTRING SRID=32632 length=878.01
row #5
    PK_UID      = 5
    Name        = 'Unknown'
    Geometry     = LINESTRING SRID=32632 length=10.05
```

```

===== table 'Regions' =====
row #1
    PK_UID      = 1
    Name        = 'VENETO'
    Geometry    = MULTIPOLYGON SRID=32632 area=646397.81
row #2
    PK_UID      = 2
    Name        = 'VENETO'
    Geometry    = MULTIPOLYGON SRID=32632 area=1290337.69
row #3
    PK_UID      = 3
    Name        = 'VENETO'
    Geometry    = MULTIPOLYGON SRID=32632 area=8784619.92
row #4
    PK_UID      = 4
    Name        = 'VENETO'
    Geometry    = MULTIPOLYGON SRID=32632 area=530524.68
row #5
    PK_UID      = 5
    Name        = 'LIGURIA'
    Geometry    = MULTIPOLYGON SRID=32632 area=5450277374.12

```

```

===== table 'Towns' =====
row #1
    PK_UID      = 1
    Name        = 'Brozolo'
    Peoples     = 435
    LocalCounc  = 1
    County      = 0
    Region      = 0
    Geometry    = POINT SRID=32632
row #2
    PK_UID      = 2
    Name        = 'Campiglione-Fenile'
    Peoples     = 1284
    LocalCounc  = 1
    County      = 0
    Region      = 0
    Geometry    = POINT SRID=32632
row #3
    PK_UID      = 3
    Name        = 'Canischio'
    Peoples     = 274
    LocalCounc  = 1
    County      = 0
    Region      = 0
    Geometry    = POINT SRID=32632
row #4
    PK_UID      = 4
    Name        = 'Cavagnolo'
    Peoples     = 2281
    LocalCounc  = 1
    County      = 0
    Region      = 0
    Geometry    = POINT SRID=32632
row #5
    PK_UID      = 5
    Name        = 'Magliano Alfieri'
    Peoples     = 1674
    LocalCounc  = 1
    County      = 0
    Region      = 0
    Geometry    = POINT SRID=32632

```

sample successfully terminated

/*

demol.c

Author: Sandro Furieri a.furieri@lgt.it

This software is provided 'as-is', without any express or implied warranty. In no event will the author be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely

```

*/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <spatialite/gaiainfig.h>
/*
these headers are required in order to support
SQLite/Spatialite
*/
#include <sqlite3.h>
#include <spatialite/gaiageo.h>
#include <spatialite.h>
int
main (int argc, char *argv[])
{
    int ret;
    sqlite3 *handle;
    sqlite3_stmt *stmt;
    gaiaGeomCollPtr geom;
    char sql[256];
    int i;
    int ic;
    char **results;
    int n_rows;
    int n_columns;
    char *err_msg = NULL;
    int len;
    char *table_name;
    char **p_geotables = NULL;
    int n_geotables = 0;
    int row_no;
    const void *blob;
    int blob_size;
    int geom_type;
    double measure;
    void *cache;
    if (argc != 2)
    {
        fprintf (stderr, "usage: %s test_db_path\n", argv[0]);
        return -1;
    }
    /*
    trying to connect the test DB:
    - this demo was designed in order to connect the standard
      TEST-2.3.SQLite sample DB
    - but you can try to use any SQLite/Spatialite DB at your will

    Please notice: we'll establish a READ ONLY connection
    */
    ret = sqlite3_open_v2 (argv[1], &handle, SQLITE_OPEN_READONLY, NULL);
    if (ret != SQLITE_OK)
    {
        printf ("cannot open '%s': %s\n", argv[1], sqlite3_errmsg (handle));
        sqlite3_close (handle);
        return -1;
    }
    /*
    VERY IMPORTANT:
    you must initialize the Spatialite extension [and related]
    BEFORE attempting to perform any other SQLite call
    =====
    Please note: starting since 4.1.0 this is completely canged:
    - a separate memory block (internal cache) is required by
      each single connection
    - allocating/freeing this block falls under the responsibility
      of the program handling the connection
    - in multithreaded programs a connection can never be share by
      different threads; the internal-cache block must be allocated
      by the same thread holding the connection
    */
    cache = spatialite_alloc_connection ();
    spatialite_init_ex (handle, cache, 0);
    /* showing the SQLite version */
    printf ("SQLite version: %s\n", sqlite3_libversion ());
    /* showing the Spatialite version */
    printf ("Spatialite version: %s\n", spatialite_version ());
    printf ("\n\n");
    /*

```

```

SQL query #1
we'll retrieve GEOMETRY tables from Spatial Metadata
we are assuming this query will return only few rows,
so this time we'll use the sqlite3_get_table() interface

this interface is very simple to use
the result set is returned as a rectangular array [rows/columns]
allocated in a temporary memory storage
so, this interface is well suited for small sized result sets,
but performs badly when accessing a large sized result set

as a side effect, each column value is returned as text, and
isn't possible at all to retrieve true column types
(INTEGER, FLOAT ...)
*/
    strcpy (sql,
        "SELECT DISTINCT f_table_name FROM geometry_columns ORDER BY 1");
    ret = sqlite3_get_table (handle, sql, &results, &n_rows, &n_columns,
        &err_msg);

    if (ret != SQLITE_OK)
    {
/* some error occurred */
        printf ("query#1 SQL error: %s\n", err_msg);
        sqlite3_free (err_msg);
        goto abort;
    }
    if (n_rows > 1)
    {
/* first row always contains column names and is meaningless in this context */
        n_geotables = n_rows;
/* allocating a dynamic pointer array to store geotable names */
        p_geotables = malloc (sizeof (char *) * n_geotables);
        for (i = 1; i <= n_rows; i++)
        {

/*
now we'll fetch one row at each time [and we have only one column to fetch]

this one is is a simplified demo; but when writing a real application
you always must check for NULL values !!!!
*/
            table_name = results[(i * n_columns) + 0];
/* and we'll store each geotable name into the dynamic pointer array */
            len = strlen (table_name);
            p_geotables[i - 1] = malloc (len + 1);
            strcpy (p_geotables[i - 1], table_name);
        }
/* we can now free the table results */
        sqlite3_free_table (results);
    }
    for (i = 0; i < n_geotables; i++)
    {
/* now we'll scan each geotable we've found in Spatial Metadata */
        printf ("===== table '%s' =====\n",
            p_geotables[i]);

/*
SQL query #2
we'll retrieve any column from the current geotable
we are assuming this query will return lots of rows,
so we have to use sqlite3_prepare_v2() interface

this interface is a more complex one, but is well
suited in order to access huge sized result sets
and true value type control is supported
*/
        sprintf (sql, "SELECT * FROM %s", p_geotables[i]);
        ret = sqlite3_prepare_v2 (handle, sql, strlen (sql), &stmt, NULL);
        if (ret != SQLITE_OK)
        {
/* some error occurred */
            printf ("query#2 SQL error: %s\n", sqlite3_errmsg (handle));
            goto abort;
        }

/*
the sqlite3_prepare_v2() call simply parses the SQL statement,
checking for syntax validity, allocating internal structs etc
but no result set row is really yet available
*/
/* we'll now save the #columns within the result set */
        n_columns = sqlite3_column_count (stmt);
        row_no = 0;
        while (1)
        {
/* this is an infinite loop, intended to fetch any row */
/* we are now trying to fetch the next available row */
            ret = sqlite3_step (stmt);
            if (ret == SQLITE_DONE)
            {

```

```

/* there are no more rows to fetch - we can stop looping */
    break;
}
if (ret == SQLITE_ROW)
{
/* ok, we've just fetched a valid row to process */
    row_no++;
    printf ("row #%d\n", row_no);
    for (ic = 0; ic < n_columns; ic++)
    {

/*
and now we'll fetch column values

for each column we'll then get:
- the column name
- a column value, that can be of type: SQLITE_NULL, SQLITE_INTEGER,
SQLITE_FLOAT, SQLITE_TEXT or SQLITE_BLOB, according to internal DB storage type
*/

        printf ("\t%-10s = ",
                sqlite3_column_name (stmt, ic));
        switch (sqlite3_column_type (stmt, ic))
        {
            case SQLITE_NULL:
                printf ("NULL");
                break;
            case SQLITE_INTEGER:
                printf ("%d", sqlite3_column_int (stmt, ic));
                break;
            case SQLITE_FLOAT:
                printf ("%1.4f",
                        sqlite3_column_double (stmt, ic));
                break;
            case SQLITE_TEXT:
                printf ("%s",
                        sqlite3_column_text (stmt, ic));
                break;
            case SQLITE_BLOB:
                blob = sqlite3_column_blob (stmt, ic);
                blob_size = sqlite3_column_bytes (stmt, ic);
/* checking if this BLOB actually is a GEOMETRY */
                geom =
                    gaiaFromSpatialLiteBlobWkb (blob,
                                                blob_size);

                if (!geom)
                {
/* for sure this one is not a GEOMETRY */
                    printf ("BLOB [%d bytes]", blob_size);
                }
                else
                {
                    geom_type = gaiaGeometryType (geom);
                    if (geom_type == GAIA_UNKNOWN)
                        printf ("EMPTY or NULL GEOMETRY");
                    else
                    {
                        char *geom_name;
                        if (geom_type == GAIA_POINT)
                            geom_name = "POINT";
                        if (geom_type == GAIA_LINESTRING)
                            geom_name = "LINESTRING";
                        if (geom_type == GAIA_POLYGON)
                            geom_name = "POLYGON";
                        if (geom_type == GAIA_MULTIPPOINT)
                            geom_name = "MULTIPOINT";
                        if (geom_type ==
                            GAIA_MULTILINESTRING)
                            geom_name = "MULTILINESTRING";
                        if (geom_type ==
                            GAIA_MULTIPOLYGON)
                            geom_name = "MULTIPOLYGON";
                        if (geom_type ==
                            GAIA_GEOMETRYCOLLECTION)
                            geom_name =
                                "GEOMETRYCOLLECTION";
                        printf ("%s SRID=%d", geom_name,
                                geom->Srid);
                        if (geom_type == GAIA_LINESTRING
                            || geom_type ==
                                GAIA_MULTILINESTRING)
                        {
                            gaiaGeomCollLength_r (cache,
                                                    geom,
                                                    &measure);
                            printf (" length=%1.2f",
                                    measure);
                        }
                    }
                }
            }
        }

#ifdef OMIT_GEOS
/* GEOS is required */
        gaiaGeomCollLength_r (cache,
                              geom,
                              &measure);
        printf (" length=%1.2f",
                measure);
#endif
    }
}
}

```

```

        printf
            (" length=?? [no GEOS support available]");
#endif /* GEOS enabled/disabled */
    }
    if (geom_type == GAIA_POLYGON ||
        geom_type ==
            GAIA_MULTIPOLYGON)
    {
#ifdef OMIT_GEOS          /* GEOS is required */
        gaiaGeomCollArea (geom,
                           &measure);
        printf (" area=%1.2f",
                 measure);
#else
        printf
            ("area=?? [no GEOS support available]");
#endif /* GEOS enabled/disabled */
    }
}

/* we have now to free the GEOMETRY */
gaiaFreeGeomColl (geom);
}
break;
};
printf ("\n");
}
if (row_no >= 5)
{
/* we'll exit the loop after the first 5 rows - this is only a demo :-) */
break;
}
}
else
{
/* some unexpected error occurred */
printf ("sqlite3_step() error: %s\n",
        sqlite3_errmsg (handle));
sqlite3_finalize (stmt);
goto abort;
}
}

/* we have now to finalize the query [memory cleanup] */
sqlite3_finalize (stmt);
printf ("\n\n");
}

/* disconnecting the test DB */
ret = sqlite3_close (handle);
if (ret != SQLITE_OK)
{
    printf ("close() error: %s\n", sqlite3_errmsg (handle));
    return -1;
}

/* freeing the internal-cache memory block */
spatialite_cleanup_ex (cache);
printf ("\n\nsample successfully terminated\n");

/* we have to free the dynamic pointer array used to store geotable names */
for (i = 0; i < n_geotables; i++)
{
/* freeing each tablename */
free (p_geotables[i]);
}
free (p_geotables);
spatialite_shutdown ();
return 0;
abort:
sqlite3_close (handle);
/* freeing the internal-cache memory block */
spatialite_cleanup_ex (cache);
if (p_geotables)
{
/* we have to free the dynamic pointer array used to store geotable names */
for (i = 0; i < n_geotables; i++)
{
/* freeing each tablename */
free (p_geotables[i]);
}
free (p_geotables);
}
spatialite_shutdown ();
return -1;
}

```

7.2 demo2.c

This is a sample C source showing how to manipulate GEOMETRY within Spatialite. It essentially follows on from the functionality shown in the demo1.c example, and covers:

- creating geometries
- exploring geometries
- querying the basic properties of a geometry

Note that this does not require a database command line argument. Here is a typical run:

```
$ ./demo2
step#1: POINT          Dimension=0 IsValid=1
                        POINT 0/1 x=1.5000 y=2.7500

step#2: LINESTRING     Dimension=1 IsValid=1
                        LINESTRING 0/1 has 5 vertices
                        vertex 0/5 x=1.0000 y=1.0000
                        vertex 1/5 x=2.0000 y=1.0000
                        vertex 2/5 x=2.0000 y=2.0000
                        vertex 3/5 x=100.0000 y=2.0000
                        vertex 4/5 x=100.0000 y=100.0000

step#3: POLYGON        Dimension=2 IsValid=1
                        POLYGON 0/1 has 2 holes
                        ExteriorRing has 5 vertices
                        vertex 0/5 x=0.0000 y=0.0000
                        vertex 1/5 x=50.0000 y=0.0000
                        vertex 2/5 x=50.0000 y=50.0000
                        vertex 3/5 x=0.0000 y=50.0000
                        vertex 4/5 x=0.0000 y=0.0000
                        InteriorRing 0/2 has 5 vertices
                        vertex 0/5 x=40.0000 y=40.0000
                        vertex 1/5 x=41.0000 y=40.0000
                        vertex 2/5 x=41.0000 y=41.0000
                        vertex 3/5 x=40.0000 y=41.0000
                        vertex 4/5 x=40.0000 y=40.0000
                        InteriorRing 1/2 has 5 vertices
                        vertex 0/5 x=30.0000 y=30.0000
                        vertex 1/5 x=31.0000 y=30.0000
                        vertex 2/5 x=31.0000 y=31.0000
                        vertex 3/5 x=30.0000 y=31.0000
                        vertex 4/5 x=30.0000 y=30.0000

step#4: MULTIPOINT     Dimension=0 IsValid=1
                        POINT 0/5 x=5.0000 y=5.0000
                        POINT 1/5 x=15.0000 y=5.0000
                        POINT 2/5 x=5.0000 y=15.0000
                        POINT 3/5 x=25.0000 y=5.0000
                        POINT 4/5 x=5.0000 y=25.0000

step#5: MULTILINESTRING Dimension=1 IsValid=1
                        LINESTRING 0/2 has 2 vertices
                        vertex 0/2 x=30.0000 y=10.0000
                        vertex 1/2 x=10.0000 y=30.0000
                        LINESTRING 1/2 has 2 vertices
                        vertex 0/2 x=40.0000 y=50.0000
                        vertex 1/2 x=50.0000 y=40.0000

step#6: MULTIPOLYGON   Dimension=2 IsValid=1
                        POLYGON 0/2 has 0 holes
                        ExteriorRing has 5 vertices
                        vertex 0/5 x=60.0000 y=60.0000
                        vertex 1/5 x=70.0000 y=60.0000
                        vertex 2/5 x=70.0000 y=70.0000
                        vertex 3/5 x=60.0000 y=70.0000
```

```

        vertex 4/5 x=60.0000 y=60.0000
    POLYGON 1/2 has 0 holes
        ExteriorRing has 5 vertices
            vertex 0/5 x=80.0000 y=80.0000
            vertex 1/5 x=90.0000 y=80.0000
            vertex 2/5 x=90.0000 y=90.0000
            vertex 3/5 x=80.0000 y=90.0000
            vertex 4/5 x=80.0000 y=80.0000

step#7: GEOMETRYCOLLECTION      Dimension=2 IsValid=1
    POINT 0/2 x=100.0000 y=100.0000
    POINT 1/2 x=100.0000 y=0.0000
    LINESTRING 0/2 has 2 vertices
        vertex 0/2 x=130.0000 y=110.0000
        vertex 1/2 x=110.0000 y=130.0000
    LINESTRING 1/2 has 2 vertices
        vertex 0/2 x=140.0000 y=150.0000
        vertex 1/2 x=150.0000 y=140.0000
    POLYGON 0/2 has 0 holes
        ExteriorRing has 5 vertices
            vertex 0/5 x=160.0000 y=160.0000
            vertex 1/5 x=170.0000 y=160.0000
            vertex 2/5 x=170.0000 y=170.0000
            vertex 3/5 x=160.0000 y=170.0000
            vertex 4/5 x=160.0000 y=160.0000
    POLYGON 1/2 has 0 holes
        ExteriorRing has 5 vertices
            vertex 0/5 x=180.0000 y=180.0000
            vertex 1/5 x=190.0000 y=180.0000
            vertex 2/5 x=190.0000 y=190.0000
            vertex 3/5 x=180.0000 y=190.0000
            vertex 4/5 x=180.0000 y=180.0000

step#8: checking WKT representations

GEOMETRYCOLLECTION(POINT(1.5 2.75))

GEOMETRYCOLLECTION(LINESTRING(1 1, 2 1, 2 2, 100 2, 100 100))

GEOMETRYCOLLECTION(POLYGON((0 0, 50 0, 50 50, 0 50, 0 0), (40 40, 41 40, 41 41, 40 41, 40 40), (30 30, 31 30, 31 31, 30 31, 30 30)))

GEOMETRYCOLLECTION(POINT(5 5), POINT(15 5), POINT(5 15), POINT(25 5), POINT(5 25))

GEOMETRYCOLLECTION(LINESTRING(30 10, 10 30), LINESTRING(40 50, 50 40))

GEOMETRYCOLLECTION(POLYGON((60 60, 70 60, 70 70, 60 70, 60 60)), POLYGON((80 80, 90 80, 90 90, 80 90, 80 80)))

GEOMETRYCOLLECTION(POINT(100 100), POINT(100 0), LINESTRING(130 110, 110 130), LINESTRING(140 150, 150 140), POINT(150 150))

/*
demo2.c

Author: Sandro Furieri a.furieri@lgt.it

This software is provided 'as-is', without any express or implied
warranty. In no event will the author be held liable for any
damages arising from the use of this software.

Permission is granted to anyone to use this software for any
purpose, including commercial applications, and to alter it and
redistribute it freely

*/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <spatialite/gaiaconfig.h>
/*
these headers are required in order to support
SQLite/Spatialite
*/
#include <geos_c.h>
#include <sqlite3.h>
#include <spatialite/gaiageo.h>
#include <spatialite.h>
static char *
geom_type (int type)
{

```



```

/* utility function returning corresponding GeometryType as a string */
static char *name = "EMPTY / NULL GEOMETRY";
if (type == GAIA_POINT)
    name = "POINT";
if (type == GAIA_LINESTRING)
    name = "LINESTRING";
if (type == GAIA_POLYGON)
    name = "POLYGON";
if (type == GAIA_MULTIPPOINT)
    name = "MULTIPOINT";
if (type == GAIA_MULTILINESTRING)
    name = "MULTILINESTRING";
if (type == GAIA_MULTIPOLYGON)
    name = "MULTIPOLYGON";
if (type == GAIA_GEOMETRYCOLLECTION)
    name = "GEOMETRYCOLLECTION";
return name;
}
static void
geometry_printout (gaiaGeomCollPtr geom)
{
/* utility function printing a generic Geometry object */
gaiaPointPtr pt;
gaiaLineStringPtr ln;
gaiaPolygonPtr pg;
gaiaRingPtr rng;
int n_pts = 0;
int n_lns = 0;
int n_pgs = 0;
int cnt;
int iv;
int ir;
double x;
double y;
/* we'll now count how many POINTs are there */
pt = geom->FirstPoint;
while (pt)
{
    n_pts++;
    pt = pt->Next;
}
/* we'll now count how many LINESTRINGs are there */
ln = geom->FirstLineString;
while (ln)
{
    n_lns++;
    ln = ln->Next;
}
/* we'll now count how many POLYGONS are there */
pg = geom->FirstPolygon;
while (pg)
{
    n_pgs++;
    pg = pg->Next;
}
if (n_pts)
{
/* printing POINTs coordinates */
cnt = 0;
pt = geom->FirstPoint;
while (pt)
{
/* we'll now scan the linked list of POINTs */
printf ("\t\t\tPOINT %d/%d x=%1.4f y=%1.4f\n",
        cnt, n_pts, pt->X, pt->Y);
cnt++;
pt = pt->Next;
}
}
if (n_lns)
{
/* printing LINESTRINGs coordinates */
cnt = 0;
ln = geom->FirstLineString;
while (ln)
{
/* we'll now scan the linked list of LINESTRINGs */
printf ("\t\t\tLINESTRING %d/%d has %d vertices\n",
        cnt, n_lns, ln->Points);
for (iv = 0; iv < ln->Points; iv++)
{
/* we'll now retrieve coordinates for each vertex */
gaiaGetPoint (ln->Coords, iv, &x, &y);
printf ("\t\t\t\tvertex %d/%d x=%1.4f y=%1.4f\n",
        iv, ln->Points, x, y);
}
cnt++;
}
}
}
}

```

```

        ln = ln->Next;
    }
}
if (n_pgs)
{
    /* printing POLYGONS coordinates */
    cnt = 0;
    pg = geom->FirstPolygon;
    while (pg)
    {
        /* we'll now scan the linked list of POLYGONS */
        printf ("\t\t\tPOLYGON %d/%d has %d hole%c\n",
            cnt, n_pgs, pg->NumInteriors,
            (pg->NumInteriors == 1) ? ' ' : 's');

        /*
        now we'll print out the Exterior ring
        [surely a POLYGON has an Exterior ring
        */
        rng = pg->Exterior;
        printf ("\t\t\tExteriorRing has %d vertices\n", rng->Points);
        for (iv = 0; iv < rng->Points; iv++)
        {
            /* we'll now retrieve coordinates for each vertex */
            gaiaGetPoint (rng->Coords, iv, &x, &y);
            printf ("\t\t\t\tvertex %d/%d x=%1.4f y=%1.4f\n",
                iv, rng->Points, x, y);
        }
        for (ir = 0; ir < pg->NumInteriors; ir++)
        {
            /*
            a POLYGON can contain an arbitrary number of Interior rings (including zero)
            */
            rng = pg->Interiors + ir;
            printf ("\t\t\t\tInteriorRing %d/%d has %d vertices\n",
                ir, pg->NumInteriors, rng->Points);
            for (iv = 0; iv < rng->Points; iv++)
            {
                /* we'll now retrieve coordinates for each vertex */
                gaiaGetPoint (rng->Coords, iv, &x, &y);
                printf ("\t\t\t\t\tvertex %d/%d x=%1.4f y=%1.4f\n",
                    iv, rng->Points, x, y);
            }
        }
        cnt++;
        pg = pg->Next;
    }
}

int
main (int argc, char *argv[])
{
    gaiaGeomCollPtr geo_pt = NULL;
    gaiaGeomCollPtr geo_ln = NULL;
    gaiaGeomCollPtr geo_pg = NULL;
    gaiaGeomCollPtr geo_mpt = NULL;
    gaiaGeomCollPtr geo_mln = NULL;
    gaiaGeomCollPtr geo_mpg = NULL;
    gaiaGeomCollPtr geo_coll = NULL;
    gaiaLinestringPtr line;
    gaiaPolygonPtr polyg;
    gaiaRingPtr ring;
    gaiaOutBuffer wkt;
    int ret;
    sqlite3 *handle;
    void *cache;

    if (argc > 1 || argv[0] == NULL)
        argc = 1;
    /* silencing stupid compiler warnings */

    /*
    this demo does not strictly require any DB connection to be established

    However you must initialize the Spatialite extension [and related]
    and you *must* establish a "fake" DB connection in order to
    properly initialize Spatialite and GEOS libraries

    */
    ret = sqlite3_open_v2 (":memory:", &handle, SQLITE_OPEN_READONLY, NULL);
    if (ret != SQLITE_OK)
    {
        printf ("cannot open '%s': %s\n", ":memory:",
            sqlite3_errmsg (handle));
        sqlite3_close (handle);
        return -1;
    }
    cache = spatialite_alloc_connection ();

```

```

    spatialite_init_ex (handle, cache, 0);
#ifdef OMIT_GEOS
    /* GEOS must be enabled */
#endif
/*
Step #1
creating and checking a POINT Geometry
*/
/* we'll allocate a Geometry object */
geo_pt = gaiaAllocGeomColl ();
/* then we insert a POINT, directly passing its coordinates */
gaiaAddPointToGeomColl (geo_pt, 1.5, 2.75);
/* now we'll print the main attributes for this geometry */
printf ("step#1: %s\tDimension=%d IsValid=%d\n",
        geom_type (gaiaGeometryType (geo_pt)),
        gaiaDimension (geo_pt), gaiaIsValid_r (cache, geo_pt));
geometry_printout (geo_pt);
/*
Step #2
creating and checking a LINESTRING Geometry
*/
geo_ln = gaiaAllocGeomColl ();
/* then we insert a LINESTRING, specifying how many vertices it contains */
line = gaiaAddLinestringToGeomColl (geo_ln, 5);
/*
we've got a pointer referencing the linestring we've just inserted
now we'll set coordinates for each vertex
*/
gaiaSetPoint (line->Coords, 0, 1.0, 1.0);
gaiaSetPoint (line->Coords, 1, 2.0, 1.0);
gaiaSetPoint (line->Coords, 2, 2.0, 2.0);
gaiaSetPoint (line->Coords, 3, 100.0, 2.0);
gaiaSetPoint (line->Coords, 4, 100.0, 100.0);
printf ("\nstep#2: %s\tDimension=%d IsValid=%d\n",
        geom_type (gaiaGeometryType (geo_ln)),
        gaiaDimension (geo_ln), gaiaIsValid_r (cache, geo_ln));
geometry_printout (geo_ln);
/*
Step #3
creating and checking a POLYGON Geometry
*/
geo_pg = gaiaAllocGeomColl ();
/*
then we insert a POLYGON, specifying:
- how many vertices have to be allocated for the Exterior Ring
- how many Interior Rings it has
*/
polyg = gaiaAddPolygonToGeomColl (geo_pg, 5, 2);
/*
we've got a pointer referencing the polygon we've just inserted
now we'll get a pointer referencing its Exterior ring
*/
ring = polyg->Exterior;
/* now we'll set coordinates for each Exterior ring vertex */
gaiaSetPoint (ring->Coords, 0, 0.0, 0.0);
gaiaSetPoint (ring->Coords, 1, 50.0, 0.0);
gaiaSetPoint (ring->Coords, 2, 50.0, 50.0);
gaiaSetPoint (ring->Coords, 3, 0.0, 50.0);
/* please note: a Ring is a CLOSED figure, so last and first vertex have to be coincident */
gaiaSetPoint (ring->Coords, 4, 0.0, 0.0);
/*
we'll now get a pointer referencing the FIRST interior ring,
specifying how vertices have to be allocated
*/
ring = gaiaAddInteriorRing (polyg, 0, 5);
/* then setting coordinates for each Interior ring vertex */
gaiaSetPoint (ring->Coords, 0, 40.0, 40.0);
gaiaSetPoint (ring->Coords, 1, 41.0, 40.0);
gaiaSetPoint (ring->Coords, 2, 41.0, 41.0);
gaiaSetPoint (ring->Coords, 3, 40.0, 41.0);
gaiaSetPoint (ring->Coords, 4, 40.0, 40.0);
/*
we'll now get a pointer referencing the SECOND interior ring,
specifying how vertices have to be allocated
*/
ring = gaiaAddInteriorRing (polyg, 1, 5);
/* then setting coordinates for each Interior ring vertex */
gaiaSetPoint (ring->Coords, 0, 30.0, 30.0);
gaiaSetPoint (ring->Coords, 1, 31.0, 30.0);
gaiaSetPoint (ring->Coords, 2, 31.0, 31.0);
gaiaSetPoint (ring->Coords, 3, 30.0, 31.0);
gaiaSetPoint (ring->Coords, 4, 30.0, 30.0);
printf ("\nstep#3: %s\tDimension=%d IsValid=%d\n",
        geom_type (gaiaGeometryType (geo_pg)),
        gaiaDimension (geo_pg), gaiaIsValid_r (cache, geo_pg));
geometry_printout (geo_pg);
/*
Step #4
creating and checking a MULTIPOINT Geometry
*/

```

```

*/
    geo_mpt = gaiaAllocGeomColl ();
/* then we'll insert some POINTs */
    gaiaAddPointToGeomColl (geo_mpt, 5.0, 5.0);
    gaiaAddPointToGeomColl (geo_mpt, 15.0, 5.0);
    gaiaAddPointToGeomColl (geo_mpt, 5.0, 15.0);
    gaiaAddPointToGeomColl (geo_mpt, 25.0, 5.0);
    gaiaAddPointToGeomColl (geo_mpt, 5.0, 25.0);
    printf ("\nstep#4: %s\tDimension=%d IsValid=%d\n",
            geom_type (gaiaGeometryType (geo_mpt)),
            gaiaDimension (geo_mpt), gaiaIsValid_r (cache, geo_mpt));
    geometry_printout (geo_mpt);
/*
Step #5
creating and checking a MULTILINESTRING Geometry
*/
    geo_mln = gaiaAllocGeomColl ();
/* then we'll insert two LINESTRINGS */
    line = gaiaAddLinestringToGeomColl (geo_mln, 2);
    gaiaSetPoint (line->Coords, 0, 30.0, 10.0);
    gaiaSetPoint (line->Coords, 1, 10.0, 30.0);
    line = gaiaAddLinestringToGeomColl (geo_mln, 2);
    gaiaSetPoint (line->Coords, 0, 40.0, 50.0);
    gaiaSetPoint (line->Coords, 1, 50.0, 40.0);
    printf ("\nstep#5: %s\tDimension=%d IsValid=%d\n",
            geom_type (gaiaGeometryType (geo_mln)),
            gaiaDimension (geo_mln), gaiaIsValid_r (cache, geo_mln));
    geometry_printout (geo_mln);
/*
Step #6
creating and checking a MULTIPOLYGON Geometry
*/
    geo_mpg = gaiaAllocGeomColl ();
/* then we'll insert two POLYGONS */
    polyg = gaiaAddPolygonToGeomColl (geo_mpg, 5, 0);
    ring = polyg->Exterior;
    gaiaSetPoint (ring->Coords, 0, 60.0, 60.0);
    gaiaSetPoint (ring->Coords, 1, 70.0, 60.0);
    gaiaSetPoint (ring->Coords, 2, 70.0, 70.0);
    gaiaSetPoint (ring->Coords, 3, 60.0, 70.0);
    gaiaSetPoint (ring->Coords, 4, 60.0, 60.0);
    polyg = gaiaAddPolygonToGeomColl (geo_mpg, 5, 0);
    ring = polyg->Exterior;
    gaiaSetPoint (ring->Coords, 0, 80.0, 80.0);
    gaiaSetPoint (ring->Coords, 1, 90.0, 80.0);
    gaiaSetPoint (ring->Coords, 2, 90.0, 90.0);
    gaiaSetPoint (ring->Coords, 3, 80.0, 90.0);
    gaiaSetPoint (ring->Coords, 4, 80.0, 80.0);
    printf ("\nstep#6: %s\tDimension=%d IsValid=%d\n",
            geom_type (gaiaGeometryType (geo_mpg)),
            gaiaDimension (geo_mpg), gaiaIsValid_r (cache, geo_mpg));
    geometry_printout (geo_mpg);
/*
Step #7
creating and checking a GEOMETRYCOLLECTION Geometry
*/
    geo_coll = gaiaAllocGeomColl ();
/* then we'll insert two POINTs */
    gaiaAddPointToGeomColl (geo_coll, 100.0, 100.0);
    gaiaAddPointToGeomColl (geo_coll, 100.0, 0.0);
/* then we'll insert two LINESTRINGS */
    line = gaiaAddLinestringToGeomColl (geo_coll, 2);
    gaiaSetPoint (line->Coords, 0, 130.0, 110.0);
    gaiaSetPoint (line->Coords, 1, 110.0, 130.0);
    line = gaiaAddLinestringToGeomColl (geo_coll, 2);
    gaiaSetPoint (line->Coords, 0, 140.0, 150.0);
    gaiaSetPoint (line->Coords, 1, 150.0, 140.0);
/* then we'll insert two POLYGONS */
    polyg = gaiaAddPolygonToGeomColl (geo_coll, 5, 0);
    ring = polyg->Exterior;
    gaiaSetPoint (ring->Coords, 0, 160.0, 160.0);
    gaiaSetPoint (ring->Coords, 1, 170.0, 160.0);
    gaiaSetPoint (ring->Coords, 2, 170.0, 170.0);
    gaiaSetPoint (ring->Coords, 3, 160.0, 170.0);
    gaiaSetPoint (ring->Coords, 4, 160.0, 160.0);
    polyg = gaiaAddPolygonToGeomColl (geo_coll, 5, 0);
    ring = polyg->Exterior;
    gaiaSetPoint (ring->Coords, 0, 180.0, 180.0);
    gaiaSetPoint (ring->Coords, 1, 190.0, 180.0);
    gaiaSetPoint (ring->Coords, 2, 190.0, 190.0);
    gaiaSetPoint (ring->Coords, 3, 180.0, 190.0);
    gaiaSetPoint (ring->Coords, 4, 180.0, 180.0);
    printf ("\nstep#7: %s\tDimension=%d IsValid=%d\n",
            geom_type (gaiaGeometryType (geo_coll)),
            gaiaDimension (geo_coll), gaiaIsValid_r (cache, geo_coll));
    geometry_printout (geo_coll);
*/

```

```

Step #8
printing each geometry as Well Known Text (WKT)
*/
    printf ("\nstep#8: checking WKT representations\n");
/* first we'll get the WKT corresponding to geometry */
    gaiaOutBufferInitialize (&wkt);
    gaiaOutWkt (&wkt, geo_pt);
/* we have to check wkt is not NULL */
    if (wkt.Error == 0 && wkt.Buffer != NULL)
    {
/* printing the WKT */
        printf ("\n%s\n", wkt.Buffer);
/* finally freeing the wkt temporary storage allocation */
        gaiaOutBufferReset (&wkt);
    }
    gaiaOutBufferInitialize (&wkt);
    gaiaOutWkt (&wkt, geo_ln);
    if (wkt.Error == 0 && wkt.Buffer != NULL)
    {
        printf ("\n%s\n", wkt.Buffer);
        gaiaOutBufferReset (&wkt);
    }
    gaiaOutBufferInitialize (&wkt);
    gaiaOutWkt (&wkt, geo_pg);
    if (wkt.Error == 0 && wkt.Buffer != NULL)
    {
        printf ("\n%s\n", wkt.Buffer);
        gaiaOutBufferReset (&wkt);
    }
    gaiaOutBufferInitialize (&wkt);
    gaiaOutWkt (&wkt, geo_mpt);
    if (wkt.Error == 0 && wkt.Buffer != NULL)
    {
        printf ("\n%s\n", wkt.Buffer);
        gaiaOutBufferReset (&wkt);
    }
    gaiaOutBufferInitialize (&wkt);
    gaiaOutWkt (&wkt, geo_mln);
    if (wkt.Error == 0 && wkt.Buffer != NULL)
    {
        printf ("\n%s\n", wkt.Buffer);
        gaiaOutBufferReset (&wkt);
    }
    gaiaOutBufferInitialize (&wkt);
    gaiaOutWkt (&wkt, geo_mpg);
    if (wkt.Error == 0 && wkt.Buffer != NULL)
    {
        printf ("\n%s\n", wkt.Buffer);
        gaiaOutBufferReset (&wkt);
    }
    gaiaOutBufferInitialize (&wkt);
    gaiaOutWkt (&wkt, geo_coll);
    if (wkt.Error == 0 && wkt.Buffer != NULL)
    {
        printf ("\n%s\n", wkt.Buffer);
        gaiaOutBufferReset (&wkt);
    }
}
#else
    printf ("no GEOS support available: skipping any test\n");
#endif /* GEOS enabled/disabled */
/*
memory cleanup
we have to destroy each object using temporary storage before exit
*/
    if (geo_pt)
        gaiaFreeGeomColl (geo_pt);
    if (geo_ln)
        gaiaFreeGeomColl (geo_ln);
    if (geo_pg)
        gaiaFreeGeomColl (geo_pg);
    if (geo_mpt)
        gaiaFreeGeomColl (geo_mpt);
    if (geo_mln)
        gaiaFreeGeomColl (geo_mln);
    if (geo_mpg)
        gaiaFreeGeomColl (geo_mpg);
    if (geo_coll)
        gaiaFreeGeomColl (geo_coll);
    sqlite3_close (handle);
    spatialite_cleanup_ex (cache);
    spatialite_shutdown ();
    return 0;
}

```

7.3 demo3.c

This is a sample C source showing how to use the SQLite / SpatiaLite Spatial Index [RTree]. It follows on from demo1.c.

The main steps in this example are:

- creating a new database
- creating a sample geo-table
- inserting 1 million rows into this table
- performing some spatial queries without Spatial Indexing
- performing the same queries using the Spatial Index

The typical output of this demo is shown below (where test.sqlite does not exist before the run).

```
$ ./demo3 test.sqlite
SQLite version: 3.7.4
SpatiaLite version: 3.0.0-beta1
```

now we are going to insert 1 million POINTs; wait, please ...

```
insert row: 25000          [elapsed time: 1.910]
insert row: 50000          [elapsed time: 4.050]
insert row: 75000          [elapsed time: 6.270]
insert row: 100000         [elapsed time: 8.460]
insert row: 125000         [elapsed time: 10.740]
insert row: 150000         [elapsed time: 12.910]
insert row: 175000         [elapsed time: 15.080]
insert row: 200000         [elapsed time: 17.350]
insert row: 225000         [elapsed time: 19.610]
insert row: 250000         [elapsed time: 21.890]
insert row: 275000         [elapsed time: 24.170]
insert row: 300000         [elapsed time: 26.380]
insert row: 325000         [elapsed time: 28.650]
insert row: 350000         [elapsed time: 30.900]
insert row: 375000         [elapsed time: 33.130]
insert row: 400000         [elapsed time: 35.340]
insert row: 425000         [elapsed time: 37.540]
insert row: 450000         [elapsed time: 39.760]
insert row: 475000         [elapsed time: 41.980]
insert row: 500000         [elapsed time: 44.220]
insert row: 525000         [elapsed time: 46.500]
insert row: 550000         [elapsed time: 48.740]
insert row: 575000         [elapsed time: 50.960]
insert row: 600000         [elapsed time: 53.190]
insert row: 625000         [elapsed time: 55.430]
insert row: 650000         [elapsed time: 57.670]
insert row: 675000         [elapsed time: 59.900]
insert row: 700000         [elapsed time: 62.130]
insert row: 725000         [elapsed time: 64.400]
insert row: 750000         [elapsed time: 66.660]
insert row: 775000         [elapsed time: 68.900]
insert row: 800000         [elapsed time: 71.140]
insert row: 825000         [elapsed time: 73.410]
insert row: 850000         [elapsed time: 75.670]
insert row: 875000         [elapsed time: 77.940]
insert row: 900000         [elapsed time: 80.230]
insert row: 925000         [elapsed time: 82.540]
insert row: 950000         [elapsed time: 84.840]
insert row: 975000         [elapsed time: 87.150]
insert row: 1000000        [elapsed time: 89.450]
```

```

performing test#0 - not using Spatial Index
Count(*) = 25          [elapsed time: 1.2700]

performing test#1 - not using Spatial Index
Count(*) = 25          [elapsed time: 1.2700]

performing test#2 - not using Spatial Index
Count(*) = 25          [elapsed time: 1.2900]

performing test#0 - using the R*Tree Spatial Index
Count(*) = 25          [elapsed time: 0.0000]

performing test#1 - using the R*Tree Spatial Index
Count(*) = 25          [elapsed time: 0.0000]

performing test#2 - using the R*Tree Spatial Index
Count(*) = 25          [elapsed time: 0.0000]

sample successfully terminated

```

Note the significant difference in elapsed time associated with use of an appropriate index.

```

/*
demo3.c

Author: Sandro Furieri a.furieri@lqt.it

This software is provided 'as-is', without any express or implied
warranty. In no event will the author be held liable for any
damages arising from the use of this software.

Permission is granted to anyone to use this software for any
purpose, including commercial applications, and to alter it and
redistribute it freely

*/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
/*
these headers are required in order to support
SQLite/Spatialite
*/
#include <sqlite3.h>
#include <spatialite/gaiageo.h>
#include <spatialite.h>
int
main (int argc, char *argv[])
{
    int ret;
    sqlite3 *handle;
    sqlite3_stmt *stmt;
    char sql[256];
    char *err_msg = NULL;
    double x;
    double y;
    int pk;
    int ix;
    int iy;
    gaiaGeomCollPtr geo = NULL;
    unsigned char *blob;
    int blob_size;
    int i;
    char **results;
    int n_rows;
    int n_columns;
    char *count;
    clock_t t0;
    clock_t t1;
    void *cache;
    if (argc != 2)
    {
        fprintf (stderr, "usage: %s test_db_path\n", argv[0]);
        return -1;
    }
    /*
    trying to connect the test DB:
    - this demo is intended to create a new, empty database
    */

```

```

ret = sqlite3_open_v2 (argv[1], &handle,
                      SQLITE_OPEN_READWRITE | SQLITE_OPEN_CREATE, NULL);
if (ret != SQLITE_OK)
{
    printf ("cannot open '%s': %s\n", argv[1], sqlite3_errmsg (handle));
    sqlite3_close (handle);
    return -1;
}
cache = spatialite_alloc_connection ();
spatialite_init_ex (handle, cache, 0);
/* showing the SQLite version */
printf ("SQLite version: %s\n", sqlite3_libversion ());
/* showing the Spatialite version */
printf ("Spatialite version: %s\n", spatialite_version ());
printf ("\n\n");
/*
we are supposing this one is an empty database,
so we have to create the Spatial Metadata
*/
strcpy (sql, "SELECT InitSpatialMetadata(1)");
ret = sqlite3_exec (handle, sql, NULL, NULL, &err_msg);
if (ret != SQLITE_OK)
{
    /* an error occurred */
    printf ("InitSpatialMetadata() error: %s\n", err_msg);
    sqlite3_free (err_msg);
    goto abort;
}

/*
now we can create the test table
for simplicity we'll define only one column, the primary key
*/
strcpy (sql, "CREATE TABLE test (");
strcat (sql, "PK INTEGER NOT NULL PRIMARY KEY)");
ret = sqlite3_exec (handle, sql, NULL, NULL, &err_msg);
if (ret != SQLITE_OK)
{
    /* an error occurred */
    printf ("CREATE TABLE 'test' error: %s\n", err_msg);
    sqlite3_free (err_msg);
    goto abort;
}

/*
... we'll add a Geometry column of POINT type to the test table
*/
strcpy (sql, "SELECT AddGeometryColumn('test', 'geom', 3003, 'POINT', 2)");
ret = sqlite3_exec (handle, sql, NULL, NULL, &err_msg);
if (ret != SQLITE_OK)
{
    /* an error occurred */
    printf ("AddGeometryColumn() error: %s\n", err_msg);
    sqlite3_free (err_msg);
    goto abort;
}

/*
and finally we'll enable this geo-column to have a Spatial Index based on R*Tree
*/
strcpy (sql, "SELECT CreateSpatialIndex('test', 'geom')");
ret = sqlite3_exec (handle, sql, NULL, NULL, &err_msg);
if (ret != SQLITE_OK)
{
    /* an error occurred */
    printf ("CreateSpatialIndex() error: %s\n", err_msg);
    sqlite3_free (err_msg);
    goto abort;
}

printf
    ("now we are going to insert 1 million POINTs; wait, please ...\n\n");
t0 = clock ();

/*
beginning a transaction

*** this step is absolutely critical ***

the SQLite engine is a TRANSACTIONAL one
the whole batch of INSERTs has to be performed as an unique transaction,
otherwise performance will be surely very poor
*/
strcpy (sql, "BEGIN");
ret = sqlite3_exec (handle, sql, NULL, NULL, &err_msg);
if (ret != SQLITE_OK)
{
    /* an error occurred */
    printf ("BEGIN error: %s\n", err_msg);
    sqlite3_free (err_msg);
    goto abort;
}

```



```

/*
preparing to populate the test table
we'll use a Prepared Statement we can reuse in order to insert each row
*/
    strcpy (sql, "INSERT INTO test (pk, geom) VALUES (?, ?)");
    ret = sqlite3_prepare_v2 (handle, sql, strlen (sql), &stmt, NULL);
    if (ret != SQLITE_OK)
    {
/* an error occurred */
        printf ("INSERT SQL error: %s\n", sqlite3_errmsg (handle));
        goto abort;
    }
    pk = 0;
    for (ix = 0; ix < 1000; ix++)
    {
        x = 1000000.0 + (ix * 10.0);
        for (iy = 0; iy < 1000; iy++)
        {
/* this double loop will insert 1 million rows into the the test table */
            y = 4000000.0 + (iy * 10.0);
            pk++;
            if ((pk % 25000) == 0)
            {
                t1 = clock ();
                printf ("insert row: %d\t\t[elapsed time: %1.3f]\n",
                    pk, (double) (t1 - t0) / CLOCKS_PER_SEC);
            }
/* preparing the geometry to insert */
            geo = gaiaAllocGeomColl ();
            geo->Srid = 3003;
            gaiaAddPointToGeomColl (geo, x, y);
/* transforming this geometry into the Spatialite BLOB format */
            gaiaToSpatialiteBlobWkb (geo, &blob, &blob_size);
/* we can now destroy the geometry object */
            gaiaFreeGeomColl (geo);
/* resetting Prepared Statement and bindings */
            sqlite3_reset (stmt);
            sqlite3_clear_bindings (stmt);
/* binding parameters to Prepared Statement */
            sqlite3_bind_int64 (stmt, 1, pk);
            sqlite3_bind_blob (stmt, 2, blob, blob_size, free);
/* performing actual row insert */
            ret = sqlite3_step (stmt);
            if (ret == SQLITE_DONE || ret == SQLITE_ROW)
            {
                ;
            }
            else
            {
/* an unexpected error occurred */
                printf ("sqlite3_step() error: %s\n",
                    sqlite3_errmsg (handle));
                sqlite3_finalize (stmt);
                goto abort;
            }
        }
    }
/* we have now to finalize the query [memory cleanup] */
    sqlite3_finalize (stmt);
/*
committing the transaction

*** this step is absolutely critical ***

if we don't confirm the still pending transaction,
any update will be lost
*/
    strcpy (sql, "COMMIT");
    ret = sqlite3_exec (handle, sql, NULL, NULL, &err_msg);
    if (ret != SQLITE_OK)
    {
/* an error occurred */
        printf ("COMMIT error: %s\n", err_msg);
        sqlite3_free (err_msg);
        goto abort;
    }
/*
now we'll optimize the table
*/
    strcpy (sql, "ANALYZE test");
    ret = sqlite3_exec (handle, sql, NULL, NULL, &err_msg);
    if (ret != SQLITE_OK)
    {
/* an error occurred */
        printf ("ANALYZE error: %s\n", err_msg);
        sqlite3_free (err_msg);
        goto abort;
    }
    for (ix = 0; ix < 3; ix++)

```

```

    {
        printf ("\nperforming test#%d - not using Spatial Index\n", ix);
    }
/*
now we'll perform the spatial query WITHOUT using the Spatial Index
we'll loop 3 times in order to avoid buffering-caching side effects
*/
    strcpy (sql, "SELECT Count(*) FROM test ");
    strcat (sql, "WHERE MbrWithin(geom, BuildMbr(");
    strcat (sql, "1000400.5, 4000400.5, ");
    strcat (sql, "1000450.5, 4000450.5))");
    t0 = clock ();
    ret = sqlite3_get_table (handle, sql, &results, &n_rows, &n_columns,
                             &err_msg);
    if (ret != SQLITE_OK)
    {
/* an error occurred */
        printf ("NoSpatialIndex SQL error: %s\n", err_msg);
        sqlite3_free (err_msg);
        goto abort;
    }
    count = "";
    for (i = 1; i <= n_rows; i++)
    {
        count = results[(i * n_columns) + 0];
    }
    t1 = clock ();
    printf ("Count(*) = %d\t\t[elapsed time: %1.4f]\n", atoi (count),
            (double) (t1 - t0) / CLOCKS_PER_SEC);
/* we can now free the table results */
    sqlite3_free_table (results);
}
for (ix = 0; ix < 3; ix++)
{
    printf ("\nperforming test#%d - using the R*Tree Spatial Index\n",
            ix);
/*
now we'll perform the spatial query USING the R*Tree Spatial Index
we'll loop 3 times in order to avoid buffering-caching side effects
*/
    strcpy (sql, "SELECT Count(*) FROM test ");
    strcat (sql, "WHERE MbrWithin(geom, BuildMbr(");
    strcat (sql, "1000400.5, 4000400.5, ");
    strcat (sql, "1000450.5, 4000450.5)) AND ROWID IN (");
    strcat (sql, "SELECT pkid FROM idx_test_geom WHERE ");
    strcat (sql, "xmin > 1000400.5 AND ");
    strcat (sql, "xmax < 1000450.5 AND ");
    strcat (sql, "ymin > 4000400.5 AND ");
    strcat (sql, "ymax < 4000450.5)");
/*
YES, this query is a very unhappy one
the idea is simply to simulate exactly the same conditions as above
*/
    t0 = clock ();
    ret = sqlite3_get_table (handle, sql, &results, &n_rows, &n_columns,
                             &err_msg);
    if (ret != SQLITE_OK)
    {
/* an error occurred */
        printf ("SpatialIndex SQL error: %s\n", err_msg);
        sqlite3_free (err_msg);
        goto abort;
    }
    count = "";
    for (i = 1; i <= n_rows; i++)
    {
        count = results[(i * n_columns) + 0];
    }
    t1 = clock ();
    printf ("Count(*) = %d\t\t[elapsed time: %1.4f]\n", atoi (count),
            (double) (t1 - t0) / CLOCKS_PER_SEC);
/* we can now free the table results */
    sqlite3_free_table (results);
}
/* disconnecting the test DB */
ret = sqlite3_close (handle);
if (ret != SQLITE_OK)
{
    printf ("close() error: %s\n", sqlite3_errmsg (handle));
    return -1;
}
spatialite_cleanup_ex (cache);
printf ("\n\nsample successfully terminated\n");
return 0;
abort:
    sqlite3_close (handle);
    spatialite_cleanup_ex (cache);
    spatialite_shutdown();

```

```
    return -1;
}
```

7.4 demo4.c

This is a sample C source showing how to use the SQLite / Spatialite Spatial Index [MbrCache]. It is very similar to demo3.c, but uses a different indexing approach

The main steps in this example are:

- creating a new database
- creating a sample geo-table
- inserting 1 million rows into this table
- performing some spatial queries without Spatial Indexing
- performing the same queries using the Spatial Index

The typical output of this demo is shown below (where test.sqlite does not exist before the run).

```
$ ./demo4 test.sqlite
SQLite version: 3.7.4
Spatialite version: 3.0.0-beta1
```

now we are going to insert 1 million POINTs; wait, please ...

```
insert row: 25000           [elapsed time: 0.370]
insert row: 50000           [elapsed time: 0.820]
insert row: 75000           [elapsed time: 1.280]
insert row: 100000          [elapsed time: 1.750]
insert row: 125000          [elapsed time: 2.210]
insert row: 150000          [elapsed time: 2.690]
insert row: 175000          [elapsed time: 3.180]
insert row: 200000          [elapsed time: 3.670]
insert row: 225000          [elapsed time: 4.210]
insert row: 250000          [elapsed time: 4.720]
insert row: 275000          [elapsed time: 5.240]
insert row: 300000          [elapsed time: 5.780]
insert row: 325000          [elapsed time: 6.330]
insert row: 350000          [elapsed time: 6.910]
insert row: 375000          [elapsed time: 7.510]
insert row: 400000          [elapsed time: 8.120]
insert row: 425000          [elapsed time: 8.750]
insert row: 450000          [elapsed time: 9.420]
insert row: 475000          [elapsed time: 10.120]
insert row: 500000          [elapsed time: 10.850]
insert row: 525000          [elapsed time: 11.610]
insert row: 550000          [elapsed time: 12.390]
insert row: 575000          [elapsed time: 13.200]
insert row: 600000          [elapsed time: 14.040]
insert row: 625000          [elapsed time: 14.900]
insert row: 650000          [elapsed time: 15.790]
insert row: 675000          [elapsed time: 16.700]
insert row: 700000          [elapsed time: 17.650]
insert row: 725000          [elapsed time: 18.620]
insert row: 750000          [elapsed time: 19.610]
insert row: 775000          [elapsed time: 20.650]
insert row: 800000          [elapsed time: 21.700]
insert row: 825000          [elapsed time: 22.760]
insert row: 850000          [elapsed time: 23.860]
insert row: 875000          [elapsed time: 25.060]
```

```

insert row: 900000          [elapsed time: 26.290]
insert row: 925000          [elapsed time: 27.480]
insert row: 950000          [elapsed time: 28.760]
insert row: 975000          [elapsed time: 30.020]
insert row: 1000000         [elapsed time: 31.280]

performing test#0 - not using Spatial Index
Count(*) = 25               [elapsed time: 1.2500]

performing test#1 - not using Spatial Index
Count(*) = 25               [elapsed time: 1.2400]

performing test#2 - not using Spatial Index
Count(*) = 25               [elapsed time: 1.2400]

performing test#0 - using the MBR cache Spatial Index
Count(*) = 25               [elapsed time: 0.0000]

performing test#1 - using the MBR cache Spatial Index
Count(*) = 25               [elapsed time: 0.0000]

performing test#2 - using the MBR cache Spatial Index
Count(*) = 25               [elapsed time: 0.0000]

sample successfully terminated

```

As for demo3.c, note the significant speed difference between the indexed and non-indexed queries.

```

/*
demo4.c

Author: Sandro Furieri a.furieri@lqt.it

This software is provided 'as-is', without any express or implied
warranty. In no event will the author be held liable for any
damages arising from the use of this software.

Permission is granted to anyone to use this software for any
purpose, including commercial applications, and to alter it and
redistribute it freely

*/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
/*
these headers are required in order to support
SQLite/Spatialite
*/
#include <sqlite3.h>
#include <spatialite/gaiageo.h>
#include <spatialite.h>
int
main (int argc, char *argv[])
{
    int ret;
    sqlite3 *handle;
    sqlite3_stmt *stmt;
    char sql[256];
    char *err_msg = NULL;
    double x;
    double y;
    int pk;
    int ix;
    int iy;
    gaiaGeomCollPtr geo = NULL;
    unsigned char *blob;
    int blob_size;
    int i;
    char **results;
    int n_rows;
    int n_columns;
    char *count;
    clock_t t0;
    clock_t t1;
    void *cache;
    if (argc != 2)
    {
        fprintf (stderr, "usage: %s test_db_path\n", argv[0]);
    }

```

```

        return -1;
    }
}

/*
trying to connect the test DB:
- this demo is intended to create a new, empty database
*/
ret = sqlite3_open_v2 (argv[1], &handle,
                      SQLITE_OPEN_READWRITE | SQLITE_OPEN_CREATE, NULL);
if (ret != SQLITE_OK)
{
    printf ("cannot open '%s': %s\n", argv[1], sqlite3_errmsg (handle));
    sqlite3_close (handle);
    return -1;
}

cache = spatialite_alloc_connection ();
spatialite_init_ex (handle, cache, 0);
/* showing the SQLite version */
printf ("SQLite version: %s\n", sqlite3_libversion ());
/* showing the Spatialite version */
printf ("Spatialite version: %s\n", spatialite_version ());
printf ("\n\n");

/*
we are supposing this one is an empty database,
so we have to create the Spatial Metadata
*/
strcpy (sql, "SELECT InitSpatialMetadata(1)");
ret = sqlite3_exec (handle, sql, NULL, NULL, &err_msg);
if (ret != SQLITE_OK)
{
    /* some error occurred */
    printf ("InitSpatialMetadata() error: %s\n", err_msg);
    sqlite3_free (err_msg);
    goto abort;
}

/*
now we can create the test table
for simplicity we'll define only one column, the primary key
*/
strcpy (sql, "CREATE TABLE test (");
strcat (sql, "PK INTEGER NOT NULL PRIMARY KEY)");
ret = sqlite3_exec (handle, sql, NULL, NULL, &err_msg);
if (ret != SQLITE_OK)
{
    /* an error occurred */
    printf ("CREATE TABLE 'test' error: %s\n", err_msg);
    sqlite3_free (err_msg);
    goto abort;
}

/*
... we'll add a Geometry column of POINT type to the test table
*/
strcpy (sql, "SELECT AddGeometryColumn('test', 'geom', 3003, 'POINT', 2)");
ret = sqlite3_exec (handle, sql, NULL, NULL, &err_msg);
if (ret != SQLITE_OK)
{
    /* an error occurred */
    printf ("AddGeometryColumn() error: %s\n", err_msg);
    sqlite3_free (err_msg);
    goto abort;
}

/*
and finally we'll enable this geo-column to have a Spatial Index based on MBR caching
*/
strcpy (sql, "SELECT CreateMbrCache('test', 'geom')");
ret = sqlite3_exec (handle, sql, NULL, NULL, &err_msg);
if (ret != SQLITE_OK)
{
    /* an error occurred */
    printf ("CreateMbrCache() error: %s\n", err_msg);
    sqlite3_free (err_msg);
    goto abort;
}

printf
    ("now we are going to insert 1 million POINTs; wait, please ...\n\n");
t0 = clock ();

/*
beginning a transaction

*** this step is absolutely critical ***

the SQLite engine is a TRANSACTIONAL one
the whole batch of INSERTs has to be performed as an unique transaction,
otherwise performance will be surely very poor
*/
strcpy (sql, "BEGIN");
ret = sqlite3_exec (handle, sql, NULL, NULL, &err_msg);
if (ret != SQLITE_OK)

```

```

    {
/* an error occurred */
        printf ("BEGIN error: %s\n", err_msg);
        sqlite3_free (err_msg);
        goto abort;
    }

/*
preparing to populate the test table
we'll use a Prepared Statement we can reuse in order to insert each row
*/
    strcpy (sql, "INSERT INTO test (pk, geom) VALUES (?, ?)");
    ret = sqlite3_prepare_v2 (handle, sql, strlen (sql), &stmt, NULL);
    if (ret != SQLITE_OK)
    {
/* an error occurred */
        printf ("INSERT SQL error: %s\n", sqlite3_errmsg (handle));
        goto abort;
    }
    pk = 0;
    for (ix = 0; ix < 1000; ix++)
    {
        x = 1000000.0 + (ix * 10.0);
        for (iy = 0; iy < 1000; iy++)
        {
/* this double loop will insert 1 million rows into the the test table */
            y = 4000000.0 + (iy * 10.0);
            pk++;
            if ((pk % 25000) == 0)
            {
                t1 = clock ();
                printf ("insert row: %d\t\t[elapsed time: %1.3f]\n",
                    pk, (double) (t1 - t0) / CLOCKS_PER_SEC);
            }
/* preparing the geometry to insert */
            geo = gaiaAllocGeomColl ();
            geo->Srid = 3003;
            gaiaAddPointToGeomColl (geo, x, y);
/* transforming this geometry into the Spatialite BLOB format */
            gaiaToSpatialiteBlobWkb (geo, &blob, &blob_size);
/* we can now destroy the geometry object */
            gaiaFreeGeomColl (geo);
/* resetting Prepared Statement and bindings */
            sqlite3_reset (stmt);
            sqlite3_clear_bindings (stmt);
/* binding parameters to Prepared Statement */
            sqlite3_bind_int64 (stmt, 1, pk);
            sqlite3_bind_blob (stmt, 2, blob, blob_size, free);
/* performing actual row insert */
            ret = sqlite3_step (stmt);
            if (ret == SQLITE_DONE || ret == SQLITE_ROW)
            {
                ;
            }
            else
            {
/* an unexpected error occurred */
                printf ("sqlite3_step() error: %s\n",
                    sqlite3_errmsg (handle));
                sqlite3_finalize (stmt);
                goto abort;
            }
        }
    }

/* we have now to finalize the query [memory cleanup] */
    sqlite3_finalize (stmt);
/*
committing the transaction

*** this step is absolutely critical ***

if we don't confirm the still pending transaction,
any update will be lost
*/
    strcpy (sql, "COMMIT");
    ret = sqlite3_exec (handle, sql, NULL, NULL, &err_msg);
    if (ret != SQLITE_OK)
    {
/* an error occurred */
        printf ("COMMIT error: %s\n", err_msg);
        sqlite3_free (err_msg);
        goto abort;
    }

/*
now we'll optimize the table
*/
    strcpy (sql, "ANALYZE test");
    ret = sqlite3_exec (handle, sql, NULL, NULL, &err_msg);
    if (ret != SQLITE_OK)
    {

```

```

/* an error occurred */
    printf ("ANALYZE error: %s\n", err_msg);
    sqlite3_free (err_msg);
    goto abort;
}
for (ix = 0; ix < 3; ix++)
{
    printf ("\nperforming test#%d - not using Spatial Index\n", ix);
}
/*
now we'll perform the spatial query WITHOUT using the Spatial Index
we'll loop 3 times in order to avoid buffering-caching side effects
*/
    strcpy (sql, "SELECT Count(*) FROM test ");
    strcat (sql, "WHERE MbrWithin(geom, BuildMbr(");
    strcat (sql, "1000400.5, 4000400.5, ");
    strcat (sql, "1000450.5, 4000450.5))");
    t0 = clock ();
    ret = sqlite3_get_table (handle, sql, &results, &n_rows, &n_columns,
                            &err_msg);
    if (ret != SQLITE_OK)
    {
/* an error occurred */
        printf ("NoSpatialIndex SQL error: %s\n", err_msg);
        sqlite3_free (err_msg);
        goto abort;
    }
    count = "";
    for (i = 1; i <= n_rows; i++)
    {
        count = results[(i * n_columns) + 0];
    }
    t1 = clock ();
    printf ("Count(*) = %d\t\t[elapsed time: %1.4f]\n", atoi (count),
            (double) (t1 - t0) / CLOCKS_PER_SEC);
/* we can now free the table results */
    sqlite3_free_table (results);
}
for (ix = 0; ix < 3; ix++)
{
    printf ("\nperforming test#%d - using the MBR cache Spatial Index\n",
            ix);
}
/*
now we'll perform the spatial query USING the MBR cache Spatial Index
we'll loop 3 times in order to avoid buffering-caching side effects
*/
    strcpy (sql, "SELECT Count(*) FROM test ");
    strcat (sql, "WHERE ROWID IN (");
    strcat (sql, "SELECT rowid FROM cache_test_geom WHERE ");
    strcat (sql,
            "mbr = FilterMbrWithin(1000400.5, 4000400.5, 1000450.5, 4000450.5))");
/*
YES, this query is a very unhappy one
the idea is simply to simulate exactly the same conditions as above
*/
    t0 = clock ();
    ret = sqlite3_get_table (handle, sql, &results, &n_rows, &n_columns,
                            &err_msg);
    if (ret != SQLITE_OK)
    {
/* an error occurred */
        printf ("SpatialIndex SQL error: %s\n", err_msg);
        sqlite3_free (err_msg);
        goto abort;
    }
    count = "";
    for (i = 1; i <= n_rows; i++)
    {
        count = results[(i * n_columns) + 0];
    }
    t1 = clock ();
    printf ("Count(*) = %d\t\t[elapsed time: %1.4f]\n", atoi (count),
            (double) (t1 - t0) / CLOCKS_PER_SEC);
/* we can now free the table results */
    sqlite3_free_table (results);
}
/* disconnecting the test DB */
ret = sqlite3_close (handle);
if (ret != SQLITE_OK)
{
    printf ("close() error: %s\n", sqlite3_errmsg (handle));
    return -1;
}
printf ("\n\nsample successfully terminated\n");
spatialite_cleanup_ex (cache);
return 0;
abort:
    sqlite3_close (handle);

```

```

    spatialite_cleanup_ex (cache);
    spatialite_shutdown();
    return -1;
}

```

7.5 demo5.c

This is a sample C source showing how to use the SpatiaLite's API [gaiaGetVectorLayersList\(\)](#), i.e. the one gathering statistic infos for Vector Layers. The typical output of this demo is shown below.

By simply specifying a DB-path demo5 will print the complete list of all Vector Layers found in that DB:

```

$ ./demo5 /home/sandro/db-4.0.sqlite
SQLite version: 3.7.11
SpatiaLite version: 4.0.0-RC2

***** VectorLayersList (mode=FAST) *****
VectorLayer: Type=BasedOnSqlTable TableName=com2011
    GeometryName=geometry SRID=23032 GeometryType=MULTIPOLYGON Dims=XY
    RowCount=8094
    ExtentMin 313360.999831 / 3933878.175118
    ExtentMax 1312106.500031 / 5220492.095518
    ReadOnly=FALSE Hidden=FALSE
VectorLayer: Type=BasedOnSqlTable TableName=prov2011
    GeometryName=geometry SRID=23032 GeometryType=MULTIPOLYGON Dims=XY
    RowCount=110
    ExtentMin 313360.999831 / 3933878.175118
    ExtentMax 1312106.500031 / 5220491.200018
    ReadOnly=FALSE Hidden=FALSE
VectorLayer: Type=BasedOnSqlTable TableName=reg2011
    GeometryName=geometry SRID=23032 GeometryType=MULTIPOLYGON Dims=XY
    RowCount=20
    ExtentMin 313360.999831 / 3933878.175118
    ExtentMax 1312106.500031 / 5220491.200018
    ReadOnly=FALSE Hidden=FALSE
VectorLayer: Type=BasedOnSqlView TableName=com_prov
    GeometryName=geometry SRID=23032 GeometryType=MULTIPOLYGON Dims=XY
    RowCount=8094
    ExtentMin 313360.999831 / 3933878.175118
    ExtentMax 1312106.500031 / 5220492.095518
    ReadOnly=FALSE Hidden=FALSE
VectorLayer: Type=BasedOnSqlView TableName=prov_reg
    GeometryName=geometry SRID=23032 GeometryType=MULTIPOLYGON Dims=XY
    RowCount=110
    ExtentMin 313360.999831 / 3933878.175118
    ExtentMax 1312106.500031 / 5220491.200018
    ReadOnly=TRUE Hidden=FALSE
VectorLayer: Type=BasedOnVirtualShape TableName=com2011a
    GeometryName=geometry SRID=23032 GeometryType=MULTIPOLYGON Dims=XY
    RowCount=8094
    ExtentMin 313360.999831 / 3933878.175118
    ExtentMax 1312106.500031 / 5220492.095518
VectorLayer: Type=BasedOnVirtualShape TableName=prov2011a
    GeometryName=geometry SRID=23032 GeometryType=MULTIPOLYGON Dims=XY
    RowCount=110
    ExtentMin 313360.999831 / 3933878.175118
    ExtentMax 1312106.500031 / 5220491.200018
VectorLayer: Type=BasedOnVirtualShape TableName=reg2011a
    GeometryName=geometry SRID=23032 GeometryType=MULTIPOLYGON Dims=XY
    RowCount=20
    ExtentMin 313360.999831 / 3933878.175118
    ExtentMax 1312106.500031 / 5220491.200018

***** VectorLayersList (mode=PRECISE) *****
VectorLayer: Type=BasedOnSqlTable TableName=com2011
    GeometryName=geometry SRID=23032 GeometryType=MULTIPOLYGON Dims=XY
    RowCount=8094

```



```

ExtentMin 313360.999831 / 3933878.175118
ExtentMax 1312106.500031 / 5220492.095518
ReadOnly=FALSE Hidden=FALSE
VectorLayer: Type=BasedOnSqlTable TableName=prov2011
GeometryName=geometry SRID=23032 GeometryType=MULTIPOLYGON Dims=XY
RowCount=110
ExtentMin 313360.999831 / 3933878.175118
ExtentMax 1312106.500031 / 5220491.200018
ReadOnly=FALSE Hidden=FALSE
VectorLayer: Type=BasedOnSqlTable TableName=reg2011
GeometryName=geometry SRID=23032 GeometryType=MULTIPOLYGON Dims=XY
RowCount=20
ExtentMin 313360.999831 / 3933878.175118
ExtentMax 1312106.500031 / 5220491.200018
ReadOnly=FALSE Hidden=FALSE
VectorLayer: Type=BasedOnSqlView TableName=com_prov
GeometryName=geometry SRID=23032 GeometryType=MULTIPOLYGON Dims=XY
RowCount=8094
ExtentMin 313360.999831 / 3933878.175118
ExtentMax 1312106.500031 / 5220492.095518
ReadOnly=FALSE Hidden=FALSE
VectorLayer: Type=BasedOnSqlView TableName=prov_reg
GeometryName=geometry SRID=23032 GeometryType=MULTIPOLYGON Dims=XY
RowCount=110
ExtentMin 313360.999831 / 3933878.175118
ExtentMax 1312106.500031 / 5220491.200018
ReadOnly=TRUE Hidden=FALSE
VectorLayer: Type=BasedOnVirtualShape TableName=com2011a
GeometryName=geometry SRID=23032 GeometryType=MULTIPOLYGON Dims=XY
RowCount=8094
ExtentMin 313360.999831 / 3933878.175118
ExtentMax 1312106.500031 / 5220492.095518
VectorLayer: Type=BasedOnVirtualShape TableName=prov2011a
GeometryName=geometry SRID=23032 GeometryType=MULTIPOLYGON Dims=XY
RowCount=110
ExtentMin 313360.999831 / 3933878.175118
ExtentMax 1312106.500031 / 5220491.200018
VectorLayer: Type=BasedOnVirtualShape TableName=reg2011a
GeometryName=geometry SRID=23032 GeometryType=MULTIPOLYGON Dims=XY
RowCount=20
ExtentMin 313360.999831 / 3933878.175118
ExtentMax 1312106.500031 / 5220491.200018

sample successfully terminated

```

By optionally specifying a Layer name demo5 will print a more detailed list for that single Layer:

```

$ ./demo5 /home/sandro/db-4.0.sqlite com2011
SQLite version: 3.7.11
SpatiaLite version: 4.0.0-RC2

***** VectorLayersList (mode=FAST) *****
VectorLayer: Type=BasedOnSqlTable TableName=com2011
GeometryName=geometry SRID=23032 GeometryType=MULTIPOLYGON Dims=XY
RowCount=8094
ExtentMin 313360.999831 / 3933878.175118
ExtentMax 1312106.500031 / 5220492.095518
ReadOnly=FALSE Hidden=FALSE
    Field #0) FieldName=PRO_COM
                IntegerValues=8094
                IntRange 1001 / 110010
    Field #1) FieldName=COD_REG
                IntegerValues=8094
                IntRange 1 / 20
    Field #2) FieldName=COD_PRO
                IntegerValues=8094
                IntRange 1 / 110
    Field #3) FieldName=NOME_COM
                TextValues=8094
                MaxSize/Length=35
    Field #4) FieldName=NOME_TED

```

```

        TextValues=8094
        MaxSize/Length=36
    Field #5) FieldName=SHAPE_Leng
        DoubleValues=8094
        DoubleRange 1566.303618 / 327044.574999
    Field #6) FieldName=SHAPE_Area
        DoubleValues=8094
        DoubleRange 120613.967719 / 1287358944.600000
    Field #7) FieldName=Geometry
        BlobValues=8094
        MaxSize/Length=222151

***** VectorLayersList (mode=PRECISE) *****
VectorLayer: Type=BasedOnSqlTable TableName=com2011
    GeometryName=geometry SRID=23032 GeometryType=MULTIPOLYGON Dims=XY
    RowCount=8094
    ExtentMin 313360.999831 / 3933878.175118
    ExtentMax 1312106.500031 / 5220492.095518
    ReadOnly=FALSE Hidden=FALSE
        Field #0) FieldName=PRO_COM
            IntegerValues=8094
            IntRange 1001 / 110010
        Field #1) FieldName=COD_REG
            IntegerValues=8094
            IntRange 1 / 20
        Field #2) FieldName=COD_PRO
            IntegerValues=8094
            IntRange 1 / 110
        Field #3) FieldName=NOME_COM
            TextValues=8094
            MaxSize/Length=35
        Field #4) FieldName=NOME_TED
            TextValues=8094
            MaxSize/Length=36
        Field #5) FieldName=SHAPE_Leng
            DoubleValues=8094
            DoubleRange 1566.303618 / 327044.574999
        Field #6) FieldName=SHAPE_Area
            DoubleValues=8094
            DoubleRange 120613.967719 / 1287358944.600000
        Field #7) FieldName=Geometry
            BlobValues=8094
            MaxSize/Length=222151

sample successfully terminated

/*
demo5.c

Author: Sandro Furieri a.furieri@lqt.it

This software is provided 'as-is', without any express or implied
warranty. In no event will the author be held liable for any
damages arising from the use of this software.

Permission is granted to anyone to use this software for any
purpose, including commercial applications, and to alter it and
redistribute it freely

*/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
/*
these headers are required in order to support
SQLite/Spatialite
*/
#include <sqlite3.h>
#include <spatialite/gaiageo.h>
#include <spatialite.h>
#define ARG_NONE 0
#define ARG_DB_PATH 1
#define ARG_TABLE 2
#define ARG_GEOMETRY 3
static void
do_print_list (gaiaVectorLayersListPtr list, int n_mode)

```

```

{
/* prints the layers list */
gaiaVectorLayerPtr lyr;
gaiaLayerAttributeFieldPtr fld;
const char *mode = "FAST";
if (n_mode == GAIA_VECTORS_LIST_OPTIMISTIC)
    mode = "OPTIMISTIC";
if (n_mode == GAIA_VECTORS_LIST_PESSIMISTIC)
    mode = "PESSIMISTIC";
printf ("\n***** VectorLayersList (mode=%s) *****\n", mode);
if (list == NULL)
{
    printf ("The VectorLayersList is empty !!!\n\n");
    return;
}
lyr = list->First;
while (lyr)
{
    /* printing the Layer Header */
    const char *lyr_type = "UnknownType";
    const char *geom_type = "UnknownType";
    const char *dims = "UnknownDims";
    switch (lyr->LayerType)
    {
        case GAIA_VECTOR_TABLE:
            lyr_type = "BasedOnSqlTable";
            break;
        case GAIA_VECTOR_VIEW:
            lyr_type = "BasedOnSqlView";
            break;
        case GAIA_VECTOR_VIRTUAL:
            lyr_type = "BasedOnVirtualShape";
            break;
    };
    switch (lyr->GeometryType)
    {
        case GAIA_VECTOR_GEOMETRY:
            geom_type = "GEOMETRY";
            break;
        case GAIA_VECTOR_POINT:
            geom_type = "POINT";
            break;
        case GAIA_VECTOR_LINESTRING:
            geom_type = "LINESTRING";
            break;
        case GAIA_VECTOR_POLYGON:
            geom_type = "POLYGON";
            break;
        case GAIA_VECTOR_MULTIPPOINT:
            geom_type = "MULTIPOINT";
            break;
        case GAIA_VECTOR_MULTILINESTRING:
            geom_type = "MULTILINESTRING";
            break;
        case GAIA_VECTOR_MULTIPOLYGON:
            geom_type = "MULTIPOLYGON";
            break;
        case GAIA_VECTOR_GEOMETRYCOLLECTION:
            geom_type = "GEOMETRYCOLLECTION";
            break;
    };
    switch (lyr->Dimensions)
    {
        case GAIA_XY:
            dims = "XY";
            break;
        case GAIA_XY_Z:
            dims = "XYZ";
            break;
        case GAIA_XY_M:
            dims = "XYM";
            break;
        case GAIA_XY_Z_M:
            dims = "XYXM";
            break;
    };
    printf ("VectorLayer: Type=%s TableName=%s\n", lyr_type,
            lyr->TableName);
    printf ("\tGeometryName=%s SRID=%d GeometryType=%s Dims=%s\n",
            lyr->GeometryName, lyr->Srid, geom_type, dims);
    if (lyr->ExtentInfos)
    {
        printf ("\tRowCount=%d\n", lyr->ExtentInfos->Count);
        printf ("\tExtentMin %f / %f\n\tExtentMax %f / %f\n",
                lyr->ExtentInfos->MinX,
                lyr->ExtentInfos->MinY, lyr->ExtentInfos->MaxX,
                lyr->ExtentInfos->MaxY);
    }
}
}

```

```

    }
    if (lyr->AuthInfos)
        printf ("\tReadOnly=%s Hidden=%s\n",
            (lyr->AuthInfos->IsReadOnly == 0) ? "FALSE" : "TRUE",
            (lyr->AuthInfos->IsHidden == 0) ? "FALSE" : "TRUE");
    fld = lyr->First;
    while (fld)
    {
        /* printing AttributeFields infos */
        printf ("\t\tField #d) FieldName=%s\n", fld->Ordinal,
            fld->AttributeName);
        printf ("\t\t\t");
        if (fld->NullValuesCount)
            printf ("NullValues=%d ", fld->NullValuesCount);
        if (fld->IntegerValuesCount)
            printf ("IntegerValues=%d ", fld->IntegerValuesCount);
        if (fld->DoubleValuesCount)
            printf ("DoubleValues=%d ", fld->DoubleValuesCount);
        if (fld->TextValuesCount)
            printf ("TextValues=%d ", fld->TextValuesCount);
        if (fld->BlobValuesCount)
            printf ("BlobValues=%d ", fld->BlobValuesCount);
        printf ("\n");
        if (fld->MaxSize)
            printf ("\t\t\tMaxSize/Length=%d\n", fld->MaxSize->MaxSize);
        if (fld->IntRange)
            printf ("\t\t\tIntRange %I64d / %I64d\n",
                fld->IntRange->MinValue, fld->IntRange->MaxValue);
        if (fld->DoubleRange)
            printf ("\t\t\tDoubleRange %f / %f\n",
                fld->DoubleRange->MinValue,
                fld->DoubleRange->MaxValue);
        fld = fld->Next;
    }
    lyr = lyr->Next;
}
printf ("\n");
}
static void
do_help ()
{
    /* printing the argument list */
    fprintf (stderr, "\n\nusage: demo5 ARGLIST\n");
    fprintf (stderr,
        "=====\n");
    fprintf (stderr, "-d or --db-path      pathname      the SpatiaLite DB path\n");
    fprintf (stderr,
        "-t or --table      table-name      the table to be checked\n");
    fprintf (stderr,
        "-g or --geometry  column_name      geometry column [optional]\n");
    fprintf (stderr, "you can specify one of the following modes:\n");
    fprintf (stderr, "-o or --optimistic      OPTIMISTIC mode\n");
    fprintf (stderr, "-p or --pessimistic     PESSIMISTIC mode\n");
}
int
main (int argc, char *argv[])
{
    int ret;
    sqlite3 *handle;
    int i;
    int next_arg = ARG_NONE;
    int mode = GAIA_VECTORS_LIST_OPTIMISTIC;
    int error = 0;
    const char *db_path = NULL;
    const char *table = NULL;
    const char *geometry = NULL;
    gaiaVectorLayersListPtr list;
    void *cache;
    for (i = 1; i < argc; i++)
    {
        /* parsing the invocation arguments */
        if (next_arg != ARG_NONE)
        {
            switch (next_arg)
            {
                case ARG_DB_PATH:
                    db_path = argv[i];
                    break;
                case ARG_TABLE:
                    table = argv[i];
                    break;
                case ARG_GEOMETRY:

```

```

        geometry = argv[i];
        break;
    };
    next_arg = ARG_NONE;
    continue;
}
if (strcasecmp (argv[i], "--help") == 0
    || strcmp (argv[i], "-h") == 0)
{
    do_help ();
    return -1;
}
if (strcasecmp (argv[i], "-d") == 0
    || strcasecmp (argv[i], "--db-path") == 0)
{
    next_arg = ARG_DB_PATH;
    continue;
}
if (strcasecmp (argv[i], "-t") == 0
    || strcmp (argv[i], "--table") == 0)
{
    next_arg = ARG_TABLE;
    continue;
}
if (strcasecmp (argv[i], "-g") == 0
    || strcmp (argv[i], "--geometry") == 0)
{
    next_arg = ARG_GEOMETRY;
    continue;
}
if (strcasecmp (argv[i], "-p") == 0
    || strcmp (argv[i], "--pessimistic") == 0)
{
    mode = GAIA_VECTORS_LIST_PESSIMISTIC;
    next_arg = ARG_NONE;
    continue;
}
if (strcasecmp (argv[i], "-o") == 0
    || strcmp (argv[i], "--optimistic") == 0)
{
    mode = GAIA_VECTORS_LIST_OPTIMISTIC;
    next_arg = ARG_NONE;
    continue;
}
fprintf (stderr, "unknown argument: %s\n", argv[i]);
error = 1;
}
if (error)
{
    do_help ();
    return -1;
}
/* checking the arguments */
if (!db_path)
{
    fprintf (stderr, "did you forget setting the --db-path argument ?\n");
    error = 1;
}
if (error)
{
    do_help ();
    return -1;
}
/*
trying to connect the test DB:
- this demo is intended to create an existing, already populated database
*/
ret = sqlite3_open_v2 (db_path, &handle,
                      SQLITE_OPEN_READWRITE | SQLITE_OPEN_CREATE, NULL);
if (ret != SQLITE_OK)
{
    printf ("cannot open '%s': %s\n", argv[1], sqlite3_errmsg (handle));
    sqlite3_close (handle);
    return -1;
}
cache = spatialite_alloc_connection ();
spatialite_init_ex (handle, cache, 0);
/* showing the SQLite version */
printf ("SQLite version: %s\n", sqlite3_libversion ());
/* showing the Spatialite version */
printf ("Spatialite version: %s\n", spatialite_version ());
printf ("\n\n");
/* listing the requested layer(s) */
list = gaiaGetVectorLayersList (handle, table, geometry, mode);
do_print_list (list, mode);
gaiaFreeVectorLayersList (list);
/* disconnecting the test DB */

```

```
ret = sqlite3_close (handle);
if (ret != SQLITE_OK)
{
    printf ("close() error: %s\n", sqlite3_errmsg (handle));
    return -1;
}
spatialite_cleanup_ex (cache);
printf ("\n\nsample successfully terminated\n");
spatialite_shutdown();
return 0;
}
```

Index

check_all_geometry_columns
 spatialite.h, [60](#)

check_all_geometry_columns_r
 spatialite.h, [61](#)

check_duplicated_rows
 spatialite.h, [62](#)

check_geometry_column
 spatialite.h, [62](#)

check_geometry_column_r
 spatialite.h, [63](#)

create_wfs_catalog
 gg_wfs.h, [552](#)

create_wfs_schema
 gg_wfs.h, [553](#)

destroy_wfs_catalog
 gg_wfs.h, [553](#)

destroy_wfs_schema
 gg_wfs.h, [554](#)

disable_tiny_point
 spatialite.h, [64](#)

dump_dbf
 spatialite.h, [64](#)

dump_dbf_ex
 spatialite.h, [65](#)

dump_dbf_ex2
 spatialite.h, [65](#)

dump_geojson
 spatialite.h, [66](#)

dump_geojson2
 spatialite.h, [67](#)

dump_geojson_ex
 spatialite.h, [68](#)

dump_kml
 spatialite.h, [69](#)

dump_kml_ex
 spatialite.h, [69](#)

dump_shapefile
 spatialite.h, [70](#)

dump_shapefile_ex
 spatialite.h, [71](#)

dump_shapefile_ex2
 spatialite.h, [71](#)

elementary_geometries
 spatialite.h, [72](#)

elementary_geometries_ex
 spatialite.h, [73](#)

elementary_geometries_ex2
 spatialite.h, [74](#)

elementary_geometries_ex3
 spatialite.h, [74](#)

enable_tiny_point
 spatialite.h, [75](#)

gaia3DDistance
 gg_advanced.h, [166](#)

gaia3dLength
 gg_advanced.h, [167](#)

gaia3DMaxDistance
 gg_advanced.h, [168](#)

gaia_create_routing
 spatialite.h, [76](#)

gaia_create_routing_nodes
 spatialite.h, [77](#)

gaia_dxf_arc, [11](#)

gaia_dxf_block, [12](#)

gaia_dxf_boundary_path, [13](#)

gaia_dxf_circle, [14](#)

gaia_dxf_extra_attr, [15](#)

gaia_dxf_hatch, [15](#)

gaia_dxf_hatch_seg, [16](#)

gaia_dxf_hole, [17](#)

gaia_dxf_insert, [18](#)

gaia_dxf_layer, [19](#)

gaia_dxf_parser, [22](#)

gaia_dxf_point, [25](#)

gaia_dxf_polyline, [26](#)

gaia_dxf_text, [27](#)

gaia_dxf_write, [28](#)

gaia_libxml2_version
 gg_xml.h, [569](#)

gaiaAddDbfField
 gg_formats.h, [455](#)

gaiaAddInteriorRing
 gg_core.h, [345](#)

gaiaAddLinestringToGeomColl
 gg_core.h, [346](#)

gaiaAddMeasure
 gg_core.h, [346](#)

gaiaAddPointToGeomColl
 gg_core.h, [347](#)

gaiaAddPointToGeomCollXYM
 gg_core.h, [347](#)

gaiaAddPointToGeomCollXYZ
 gg_core.h, [348](#)

gaiaAddPointToGeomCollXYZM
 gg_core.h, [348](#)

gaiaAddPolygonToGeomColl
 gg_core.h, [349](#)

- gaiaAddRingToPolyg
 - gg_core.h, [349](#)
- gaiaAllocDbf
 - gg_formats.h, [456](#)
- gaiaAllocDbfField
 - gg_formats.h, [457](#)
- gaiaAllocDbfList
 - gg_formats.h, [457](#)
- gaiaAllocDynamicLine
 - gg_dynamic.h, [439](#)
- gaiaAllocGeomColl
 - gg_core.h, [350](#)
- gaiaAllocGeomCollXYM
 - gg_core.h, [350](#)
- gaiaAllocGeomCollXYZ
 - gg_core.h, [351](#)
- gaiaAllocGeomCollXYZM
 - gg_core.h, [351](#)
- gaiaAllocLinestring
 - gg_core.h, [352](#)
- gaiaAllocLinestringXYM
 - gg_core.h, [352](#)
- gaiaAllocLinestringXYZ
 - gg_core.h, [353](#)
- gaiaAllocLinestringXYZM
 - gg_core.h, [354](#)
- gaiaAllocPoint
 - gg_core.h, [354](#)
- gaiaAllocPointXYM
 - gg_core.h, [355](#)
- gaiaAllocPointXYZ
 - gg_core.h, [355](#)
- gaiaAllocPointXYZM
 - gg_core.h, [357](#)
- gaiaAllocPolygon
 - gg_core.h, [358](#)
- gaiaAllocPolygonXYM
 - gg_core.h, [358](#)
- gaiaAllocPolygonXYZ
 - gg_core.h, [359](#)
- gaiaAllocPolygonXYZM
 - gg_core.h, [359](#)
- gaiaAllocRing
 - gg_core.h, [360](#)
- gaiaAllocRingXYM
 - gg_core.h, [360](#)
- gaiaAllocRingXYZ
 - gg_core.h, [361](#)
- gaiaAllocRingXYZM
 - gg_core.h, [362](#)
- gaiaAllocShapefile
 - gg_formats.h, [458](#)
- gaiaAppendPointMToDynamicLine
 - gg_dynamic.h, [439](#)
- gaiaAppendPointToDynamicLine
 - gg_dynamic.h, [440](#)
- gaiaAppendPointZMToDynamicLine
 - gg_dynamic.h, [440](#)
- gaiaAppendPointZToDynamicLine
 - gg_dynamic.h, [440](#)
- gaiaAppendToOutBuffer
 - gg_formats.h, [458](#)
- gaiaAsEncodedPolyLine
 - gg_advanced.h, [168](#)
- gaiaAsX3D
 - gg_advanced.h, [169](#)
- gaiaAttributeFieldDoubleRangeInfos, [28](#)
- gaiaAttributeFieldDoubleRangePtr
 - gg_structs.h, [545](#)
- gaiaAttributeFieldIntRangeInfos, [29](#)
- gaiaAttributeFieldIntRangePtr
 - gg_structs.h, [545](#)
- gaiaAttributeFieldMaxSizeInfos, [29](#)
- gaiaAttributeFieldMaxSizePtr
 - gg_structs.h, [545](#)
- gaiaaux.h
 - gaiaCleanSqlString, [118](#)
 - gaiaConvertCharset, [118](#)
 - gaiaConvertToDMS, [120](#)
 - gaiaConvertToDMSex, [120](#)
 - gaiaConvertToUTF8, [122](#)
 - gaiaCreateMD5Checksum, [123](#)
 - gaiaCreateUTF8Converter, [123](#)
 - gaiaDecodeURL, [123](#)
 - gaiaDequotedSql, [124](#)
 - gaiaDirNameFromPath, [125](#)
 - gaiaDoubleQuotedSql, [125](#)
 - gaiaEncodeURL, [126](#)
 - gaiaFileExtFromPath, [126](#)
 - gaiaFileNameFromPath, [127](#)
 - gaiaFinalizeMD5Checksum, [127](#)
 - gaiaFreeMD5Checksum, [128](#)
 - gaiaFreeUTF8Converter, [128](#)
 - gaiaFullFileNameFromPath, [129](#)
 - gaiaGetLocaleCharset, [129](#)
 - gaiaIllegalSqlName, [129](#)
 - gaiaInsertIntoSqlLog, [130](#)
 - gaialsReservedSqliteName, [130](#)
 - gaialsReservedSqlName, [131](#)
 - gaiaParseDMS, [131](#)
 - gaiaQuotedSql, [132](#)
 - gaiaRemoveExtraSpaces, [132](#)
 - gaiaSingleQuotedSql, [133](#)
 - gaiaUpdateMD5Checksum, [133](#)
 - gaiaUpdateSqlLog, [134](#)
- gaiaAzimuth
 - gg_advanced.h, [170](#)
- gaiaBoundary
 - gg_advanced.h, [170](#)
- gaiaBoundary_r
 - gg_advanced.h, [171](#)
- gaiaBuildCircleMbr
 - gg_mbr.h, [522](#)
- gaiaBuildFilterMbr
 - gg_mbr.h, [522](#)
- gaiaBuildMbr

gg_mbr.h, 523
gaiaCastGeomCollToXY
gg_core.h, 362
gaiaCastGeomCollToXYM
gg_core.h, 363
gaiaCastGeomCollToXYMnoData
gg_core.h, 363
gaiaCastGeomCollToXYZ
gg_core.h, 364
gaiaCastGeomCollToXYZM
gg_core.h, 364
gaiaCastGeomCollToXYZMnoData
gg_core.h, 366
gaiaCastGeomCollToXYZnoData
gg_core.h, 366
gaiaCheckClockwise
gg_core.h, 367
gaiaCheckCounterClockwise
gg_core.h, 367
gaiaCleanSqlString
gaiaaux.h, 118
gaiaClockwise
gg_core.h, 368
gaiaCloneDbfEntity
gg_formats.h, 459
gaiaCloneDbfField
gg_formats.h, 459
gaiaCloneDynamicLine
gg_dynamic.h, 441
gaiaCloneGeomColl
gg_core.h, 368
gaiaCloneGeomCollLinestrings
gg_core.h, 369
gaiaCloneGeomCollPoints
gg_core.h, 369
gaiaCloneGeomCollPolygons
gg_core.h, 370
gaiaCloneGeomCollSpecial
gg_core.h, 370
gaiaCloneLinestring
gg_core.h, 371
gaiaCloneLinestringSpecial
gg_core.h, 372
gaiaClonePolygon
gg_core.h, 372
gaiaClonePolygonSpecial
gg_core.h, 373
gaiaCloneRing
gg_core.h, 373
gaiaCloneRingSpecial
gg_core.h, 374
gaiaCloneValue
gg_formats.h, 460
gaiaConcaveHull
gg_advanced.h, 172
gaiaConcaveHull_r
gg_advanced.h, 173
gaiaConstrainedDelaunayTriangulation
gg_advanced.h, 173
gaiaConstrainedDelaunayTriangulation_r
gg_advanced.h, 174
gaiaConvertCharset
gaiaaux.h, 118
gaiaConvertLength
gg_core.h, 374
gaiaConvertToDMS
gaiaaux.h, 120
gaiaConvertToDMSEX
gaiaaux.h, 120
gaiaConvertToUTF8
gaiaaux.h, 122
gaiaConvexHull
gg_advanced.h, 175
gaiaConvexHull_r
gg_advanced.h, 175
gaiaCopyLinestringCoords
gg_core.h, 375
gaiaCopyLinestringCoordsEx
gg_core.h, 375
gaiaCopyLinestringCoordsReverse
gg_core.h, 376
gaiaCopyRingCoords
gg_core.h, 376
gaiaCopyRingCoordsEx
gg_core.h, 377
gaiaCopyRingCoordsReverse
gg_core.h, 377
gaiaCreateDxfParser
gg_dxf.h, 426
gaiaCreateDynamicLine
gg_dynamic.h, 441
gaiaCreateMD5Checksum
gaiaaux.h, 123
gaiaCreateMetaCatalogTables
spatialite.h, 77
gaiaCreatePolygon
gg_core.h, 378
gaiaCreateUTF8Converter
gaiaaux.h, 123
gaiaCriticalPointFromGEOSmsg
gg_advanced.h, 176
gaiaCriticalPointFromGEOSmsg_r
gg_advanced.h, 176
gaiaCurvosityIndex
gg_core.h, 378
gaiaCutter
spatialite.h, 78
gaiaDbfFieldStruct, 30
gaiaDbfListPtr
gg_structs.h, 546
gaiaDbfListStruct, 31
gaiaDbfPtr
gg_structs.h, 546
gaiaDbfStruct, 32
gaiaDecodeURL
gaiaaux.h, 123

gaiaDegsToRads
 [gg_advanced.h, 177](#)
gaiaDelaunayTriangulation
 [gg_advanced.h, 177](#)
gaiaDelaunayTriangulation_r
 [gg_advanced.h, 178](#)
gaiaDequotedSql
 [gaiaaux.h, 124](#)
gaiaDestroyDxfParser
 [gg_dxf.h, 426](#)
gaiaDimension
 [gg_core.h, 379](#)
gaiaDirNameFromPath
 [gaiaaux.h, 125](#)
gaiaDissolvePoints
 [gg_core.h, 379](#)
gaiaDissolveSegments
 [gg_core.h, 380](#)
gaiaDoubleQuotedSql
 [gaiaaux.h, 125](#)
gaiaDrapeLine
 [gg_advanced.h, 179](#)
gaiaDrapeLineExceptions
 [gg_advanced.h, 180](#)
gaiaDropTable
 [spatialite.h, 79](#)
gaiaDropTable5
 [spatialite.h, 79](#)
gaiaDropTableEx
 [spatialite.h, 80](#)
gaiaDropTableEx2
 [spatialite.h, 81](#)
gaiaDropTableEx3
 [spatialite.h, 81](#)
gaiaDxfArcPtr
 [gg_dxf.h, 422](#)
gaiaDxfBlockPtr
 [gg_dxf.h, 422](#)
gaiaDxfBoundaryPathPtr
 [gg_dxf.h, 423](#)
gaiaDxfCirclePtr
 [gg_dxf.h, 423](#)
gaiaDxfExtraAttrPtr
 [gg_dxf.h, 423](#)
gaiaDxfHatchPtr
 [gg_dxf.h, 423](#)
gaiaDxfHatchSegmPtr
 [gg_dxf.h, 424](#)
gaiaDxfHolePtr
 [gg_dxf.h, 424](#)
gaiaDxfInsertPtr
 [gg_dxf.h, 424](#)
gaiaDxfLayerPtr
 [gg_dxf.h, 424](#)
gaiaDxfParserPtr
 [gg_dxf.h, 425](#)
gaiaDxfPointPtr
 [gg_dxf.h, 425](#)
gaiaDxfPolylinePtr
 [gg_dxf.h, 425](#)
gaiaDxfTextPtr
 [gg_dxf.h, 425](#)
gaiaDxfWriteEndSection
 [gg_dxf.h, 427](#)
gaiaDxfWriteEntities
 [gg_dxf.h, 427](#)
gaiaDxfWriteFooter
 [gg_dxf.h, 428](#)
gaiaDxfWriteGeometry
 [gg_dxf.h, 428](#)
gaiaDxfWriteHeader
 [gg_dxf.h, 429](#)
gaiaDxfWriteLayer
 [gg_dxf.h, 429](#)
gaiaDxfWriteLine
 [gg_dxf.h, 430](#)
gaiaDxfWritePoint
 [gg_dxf.h, 430](#)
gaiaDxfWriteRing
 [gg_dxf.h, 431](#)
gaiaDxfWriterInit
 [gg_dxf.h, 432](#)
gaiaDxfWriteTables
 [gg_dxf.h, 432](#)
gaiaDxfWriteText
 [gg_dxf.h, 433](#)
gaiaDynamicLineDeletePoint
 [gg_dynamic.h, 442](#)
gaiaDynamicLineFindByCoords
 [gg_dynamic.h, 442](#)
gaiaDynamicLineFindByPos
 [gg_dynamic.h, 443](#)
gaiaDynamicLineInsertAfter
 [gg_dynamic.h, 443](#)
gaiaDynamicLineInsertBefore
 [gg_dynamic.h, 444](#)
gaiaDynamicLineJoinAfter
 [gg_dynamic.h, 444](#)
gaiaDynamicLineJoinBefore
 [gg_dynamic.h, 445](#)
gaiaDynamicLinePtr
 [gg_structs.h, 546](#)
gaiaDynamicLineSplitAfter
 [gg_dynamic.h, 446](#)
gaiaDynamicLineSplitBefore
 [gg_dynamic.h, 446](#)
gaiaDynamicLineStruct, [33](#)
gaiaEllipseParams
 [gg_core.h, 381](#)
gaiaEllipsoidAzimuth
 [gg_advanced.h, 180](#)
gaiaEncodeURL
 [gaiaaux.h, 126](#)
gaiaEndianArch
 [gg_formats.h, 460](#)
gaiaEnsureClosedRings

- gg_core.h, 381
- gaiaEwkbGetLinestring
 - gg_formats.h, 461
- gaiaEwkbGetMultiGeometry
 - gg_formats.h, 461
- gaiaEwkbGetPoint
 - gg_formats.h, 462
- gaiaEwkbGetPolygon
 - gg_formats.h, 463
- gaiaexif.h
 - gaiaExifTagGetByteValue, 138
 - gaiaExifTagGetDoubleValue, 139
 - gaiaExifTagGetFloatValue, 139
 - gaiaExifTagGetHumanReadable, 140
 - gaiaExifTagGetId, 140
 - gaiaExifTagGetLongValue, 141
 - gaiaExifTagGetName, 141
 - gaiaExifTagGetNumValues, 142
 - gaiaExifTagGetRational1Value, 142
 - gaiaExifTagGetRational2Value, 142
 - gaiaExifTagGetRationalValue, 143
 - gaiaExifTagGetShortValue, 143
 - gaiaExifTagGetSignedLongValue, 144
 - gaiaExifTagGetSignedRational1Value, 144
 - gaiaExifTagGetSignedRational2Value, 145
 - gaiaExifTagGetSignedRationalValue, 145
 - gaiaExifTagGetSignedShortValue, 146
 - gaiaExifTagGetStringValue, 146
 - gaiaExifTagGetValueType, 147
 - gaiaExifTagListPtr, 138
 - gaiaExifTagPtr, 138
 - gaiaExifTagsFree, 147
 - gaiaGetExifGpsTagById, 148
 - gaiaGetExifTagById, 148
 - gaiaGetExifTagByName, 149
 - gaiaGetExifTagByPos, 149
 - gaiaGetExifTags, 150
 - gaiaGetExifTagsCount, 150
 - gaiaGetGpsCoords, 151
 - gaiaGetGpsLatLong, 151
 - gaiaGuessBlobType, 152
 - gaiaIsExifGpsTag, 152
- gaiaExifTagGetByteValue
 - gaiaexif.h, 138
- gaiaExifTagGetDoubleValue
 - gaiaexif.h, 139
- gaiaExifTagGetFloatValue
 - gaiaexif.h, 139
- gaiaExifTagGetHumanReadable
 - gaiaexif.h, 140
- gaiaExifTagGetId
 - gaiaexif.h, 140
- gaiaExifTagGetLongValue
 - gaiaexif.h, 141
- gaiaExifTagGetName
 - gaiaexif.h, 141
- gaiaExifTagGetNumValues
 - gaiaexif.h, 142
- gaiaExifTagGetRational1Value
 - gaiaexif.h, 142
- gaiaExifTagGetRational2Value
 - gaiaexif.h, 142
- gaiaExifTagGetRationalValue
 - gaiaexif.h, 143
- gaiaExifTagGetShortValue
 - gaiaexif.h, 143
- gaiaExifTagGetSignedLongValue
 - gaiaexif.h, 144
- gaiaExifTagGetSignedRational1Value
 - gaiaexif.h, 144
- gaiaExifTagGetSignedRational2Value
 - gaiaexif.h, 145
- gaiaExifTagGetSignedRationalValue
 - gaiaexif.h, 145
- gaiaExifTagGetSignedShortValue
 - gaiaexif.h, 146
- gaiaExifTagGetStringValue
 - gaiaexif.h, 146
- gaiaExifTagGetValueType
 - gaiaexif.h, 147
- gaiaExifTagListPtr
 - gaiaexif.h, 138
- gaiaExifTagListStruct, 34
- gaiaExifTagPtr
 - gaiaexif.h, 138
- gaiaExifTagsFree
 - gaiaexif.h, 147
- gaiaExifTagStruct, 35
- gaiaExport16
 - gg_formats.h, 464
- gaiaExport32
 - gg_formats.h, 464
- gaiaExport64
 - gg_formats.h, 465
- gaiaExportDxf
 - gg_dxf.h, 433
- gaiaExportF32
 - gg_formats.h, 465
- gaiaExportI64
 - gg_formats.h, 466
- gaiaExportU32
 - gg_formats.h, 466
- gaiaExtractLinestringsFromGeomColl
 - gg_core.h, 382
- gaiaExtractPointsFromGeomColl
 - gg_core.h, 382
- gaiaExtractPolygonsFromGeomColl
 - gg_core.h, 383
- gaiaFileExtFromPath
 - gaiaaux.h, 126
- gaiaFileNameFromPath
 - gaiaaux.h, 127
- gaiaFinalizeMD5Checksum
 - gaiaaux.h, 127
- gaiaFlushDbfHeader
 - gg_formats.h, 467

- gaiaFlushShpHeaders
 - gg_formats.h, [467](#)
- gaiaFrechetDistance
 - gg_advanced.h, [181](#)
- gaiaFrechetDistance_r
 - gg_advanced.h, [182](#)
- gaiaFrechetDistanceDensify
 - gg_advanced.h, [182](#)
- gaiaFrechetDistanceDensify_r
 - gg_advanced.h, [183](#)
- gaiaFree
 - gg_core.h, [384](#)
- gaiaFreeDbf
 - gg_formats.h, [468](#)
- gaiaFreeDbfField
 - gg_formats.h, [468](#)
- gaiaFreeDbfList
 - gg_formats.h, [468](#)
- gaiaFreeDynamicLine
 - gg_dynamic.h, [447](#)
- gaiaFreeGeomColl
 - gg_core.h, [384](#)
- gaiaFreeLinestring
 - gg_core.h, [385](#)
- gaiaFreeMD5Checksum
 - gaiaaux.h, [128](#)
- gaiaFreePoint
 - gg_core.h, [385](#)
- gaiaFreePolygon
 - gg_core.h, [385](#)
- gaiaFreeRing
 - gg_core.h, [386](#)
- gaiaFreeShapefile
 - gg_formats.h, [469](#)
- gaiaFreeUTF8Converter
 - gaiaaux.h, [128](#)
- gaiaFreeValue
 - gg_formats.h, [469](#)
- gaiaFreeVectorLayersList
 - spatialite.h, [82](#)
- gaiaFromEWKB
 - gg_formats.h, [470](#)
- gaiaFromFgf
 - gg_formats.h, [470](#)
- gaiaFromGeos_XY
 - gg_advanced.h, [184](#)
- gaiaFromGeos_XY_r
 - gg_advanced.h, [185](#)
- gaiaFromGeos_XYM
 - gg_advanced.h, [185](#)
- gaiaFromGeos_XYM_r
 - gg_advanced.h, [186](#)
- gaiaFromGeos_XYZ
 - gg_advanced.h, [187](#)
- gaiaFromGeos_XYZ_r
 - gg_advanced.h, [187](#)
- gaiaFromGeos_XYZM
 - gg_advanced.h, [188](#)
- gaiaFromGeos_XYZM_r
 - gg_advanced.h, [189](#)
- gaiaFromSpatiaLiteBlobMbr
 - gg_mbr.h, [524](#)
- gaiaFromSpatiaLiteBlobWkb
 - gg_formats.h, [471](#)
- gaiaFromSpatiaLiteBlobWkbEx
 - gg_formats.h, [471](#)
- gaiaFromTWKB
 - gg_advanced.h, [189](#)
- gaiaFromWkb
 - gg_formats.h, [472](#)
- gaiaFullFileNameFromPath
 - gaiaaux.h, [129](#)
- gaiaGeodesicArcLength
 - gg_core.h, [386](#)
- gaiaGeodesicArea
 - gg_advanced.h, [190](#)
- gaiaGeodesicDistance
 - gg_core.h, [387](#)
- gaiaGeodesicTotalLength
 - gg_core.h, [388](#)
- gaiaGeoHash
 - gg_advanced.h, [191](#)
- gaiaGeomCollArea
 - gg_advanced.h, [191](#)
- gaiaGeomCollArea_r
 - gg_advanced.h, [192](#)
- gaiaGeomCollBuffer
 - gg_advanced.h, [193](#)
- gaiaGeomCollBuffer_r
 - gg_advanced.h, [193](#)
- gaiaGeomCollCentroid
 - gg_advanced.h, [194](#)
- gaiaGeomCollCentroid_r
 - gg_advanced.h, [195](#)
- gaiaGeomCollContains
 - gg_advanced.h, [195](#)
- gaiaGeomCollContains_r
 - gg_advanced.h, [196](#)
- gaiaGeomCollCoveredBy
 - gg_advanced.h, [197](#)
- gaiaGeomCollCoveredBy_r
 - gg_advanced.h, [197](#)
- gaiaGeomCollCovers
 - gg_advanced.h, [198](#)
- gaiaGeomCollCovers_r
 - gg_advanced.h, [199](#)
- gaiaGeomCollCrosses
 - gg_advanced.h, [199](#)
- gaiaGeomCollCrosses_r
 - gg_advanced.h, [200](#)
- gaiaGeomCollDisjoint
 - gg_advanced.h, [201](#)
- gaiaGeomCollDisjoint_r
 - gg_advanced.h, [201](#)
- gaiaGeomCollDistance
 - gg_advanced.h, [202](#)

gaiaGeomCollDistance_r
 gg_advanced.h, 203

gaiaGeomCollEquals
 gg_advanced.h, 204

gaiaGeomCollEquals_r
 gg_advanced.h, 204

gaiaGeomCollIntersects
 gg_advanced.h, 205

gaiaGeomCollIntersects_r
 gg_advanced.h, 206

gaiaGeomCollLength
 gg_advanced.h, 206

gaiaGeomCollLength_r
 gg_advanced.h, 207

gaiaGeomCollLengthOrPerimeter
 gg_advanced.h, 208

gaiaGeomCollLengthOrPerimeter_r
 gg_advanced.h, 208

gaiaGeomCollOverlaps
 gg_advanced.h, 209

gaiaGeomCollOverlaps_r
 gg_advanced.h, 210

gaiaGeomCollPreparedContains
 gg_advanced.h, 210

gaiaGeomCollPreparedCoveredBy
 gg_advanced.h, 211

gaiaGeomCollPreparedCovers
 gg_advanced.h, 212

gaiaGeomCollPreparedCrosses
 gg_advanced.h, 213

gaiaGeomCollPreparedDisjoint
 gg_advanced.h, 214

gaiaGeomCollPreparedDistance
 gg_advanced.h, 214

gaiaGeomCollPreparedDistanceWithin
 gg_advanced.h, 215

gaiaGeomCollPreparedIntersects
 gg_advanced.h, 216

gaiaGeomCollPreparedOverlaps
 gg_advanced.h, 217

gaiaGeomCollPreparedTouches
 gg_advanced.h, 218

gaiaGeomCollPreparedWithin
 gg_advanced.h, 218

gaiaGeomCollPtr
 gg_structs.h, 546

gaiaGeomCollRelate
 gg_advanced.h, 219

gaiaGeomCollRelate_r
 gg_advanced.h, 220

gaiaGeomCollRelateBoundaryNodeRule
 gg_advanced.h, 221

gaiaGeomCollRelateBoundaryNodeRule_r
 gg_advanced.h, 221

gaiaGeomCollSimplify
 gg_advanced.h, 222

gaiaGeomCollSimplify_r
 gg_advanced.h, 223

gaiaGeomCollSimplifyPreserveTopology
 gg_advanced.h, 224

gaiaGeomCollSimplifyPreserveTopology_r
 gg_advanced.h, 224

gaiaGeomCollStruct, 36

gaiaGeomCollTouches
 gg_advanced.h, 225

gaiaGeomCollTouches_r
 gg_advanced.h, 226

gaiaGeomCollWithin
 gg_advanced.h, 226

gaiaGeomCollWithin_r
 gg_advanced.h, 227

gaiaGeometryAliasType
 gg_core.h, 389

gaiaGeometryDifference
 gg_advanced.h, 228

gaiaGeometryDifference_r
 gg_advanced.h, 228

gaiaGeometryIntersection
 gg_advanced.h, 229

gaiaGeometryIntersection_r
 gg_advanced.h, 230

gaiaGeometrySymDifference
 gg_advanced.h, 230

gaiaGeometrySymDifference_r
 gg_advanced.h, 232

gaiaGeometryType
 gg_core.h, 389

gaiaGeometryUnion
 gg_advanced.h, 233

gaiaGeometryUnion_r
 gg_advanced.h, 233

gaiaGeosConcaveHull
 gg_advanced.h, 234

gaiaGeosConcaveHull_r
 gg_advanced.h, 235

gaiaGeosDensify
 gg_advanced.h, 236

gaiaGeosDensify_r
 gg_advanced.h, 236

gaiaGeosMakeValid
 gg_advanced.h, 237

gaiaGeosMakeValid_r
 gg_advanced.h, 238

gaiaGetExifGpsTagById
 gaiaexif.h, 148

gaiaGetExifTagById
 gaiaexif.h, 148

gaiaGetExifTagByName
 gaiaexif.h, 149

gaiaGetExifTagByPos
 gaiaexif.h, 149

gaiaGetExifTags
 gaiaexif.h, 150

gaiaGetExifTagsCount
 gaiaexif.h, 150

gaiaGetGeosAuxErrorMsg

- gg_advanced.h, 238
- gaiaGetGeosAuxErrorMsg_r
 - gg_advanced.h, 239
- gaiaGetGeosErrorMsg
 - gg_advanced.h, 240
- gaiaGetGeosErrorMsg_r
 - gg_advanced.h, 240
- gaiaGetGeosWarningMsg
 - gg_advanced.h, 241
- gaiaGetGeosWarningMsg_r
 - gg_advanced.h, 241
- gaiaGetGpsCoords
 - gaiaexif.h, 151
- gaiaGetGpsLatLong
 - gaiaexif.h, 151
- gaiaGetLayerExtent
 - spatialite.h, 82
- gaiaGetLocaleCharset
 - gaiaaux.h, 129
- gaiaGetMbrMaxX
 - gg_mbr.h, 524
- gaiaGetMbrMaxY
 - gg_mbr.h, 525
- gaiaGetMbrMinX
 - gg_mbr.h, 525
- gaiaGetMbrMinY
 - gg_mbr.h, 526
- gaiaGetPoint
 - gg_const.h, 333
- gaiaGetPointOnSurface
 - gg_advanced.h, 242
- gaiaGetPointOnSurface_r
 - gg_advanced.h, 243
- gaiaGetPointXYM
 - gg_const.h, 334
- gaiaGetPointXYZ
 - gg_const.h, 334
- gaiaGetPointXYZM
 - gg_const.h, 335
- gaiaGetProjDatabasePath
 - gg_advanced.h, 243
- gaiaGetProjErrorMsg_r
 - gg_advanced.h, 244
- gaiaGetProjString
 - gg_advanced.h, 244
- gaiaGetProjWKT
 - gg_advanced.h, 245
- gaiaGetRtTopoErrorMsg
 - gg_advanced.h, 246
- gaiaGetRtTopoWarningMsg
 - gg_advanced.h, 246
- gaiaGetVectorLayersList
 - spatialite.h, 83
- gaiaGreatCircleDistance
 - gg_core.h, 390
- gaiaGreatCircleTotalLength
 - gg_core.h, 391
- gaiaGuessBlobType
 - gaiaexif.h, 152
- gaiaGuessSridFromWKT
 - gg_advanced.h, 247
- gaiaHausdorffDistance
 - gg_advanced.h, 247
- gaiaHausdorffDistance_r
 - gg_advanced.h, 248
- gaiaHausdorffDistanceDensify
 - gg_advanced.h, 249
- gaiaHausdorffDistanceDensify_r
 - gg_advanced.h, 249
- gaiaHexagonalGrid
 - gg_advanced.h, 250
- gaiaHexagonalGrid_r
 - gg_advanced.h, 251
- gaiaHilbertCode
 - gg_advanced.h, 252
- gaiaHilbertCode_r
 - gg_advanced.h, 252
- gaiaIllegalSqlName
 - gaiaaux.h, 129
- gaiaImport16
 - gg_formats.h, 473
- gaiaImport32
 - gg_formats.h, 473
- gaiaImport64
 - gg_formats.h, 474
- gaiaImportF32
 - gg_formats.h, 474
- gaiaImportI64
 - gg_formats.h, 475
- gaiaImportU32
 - gg_formats.h, 476
- gaiaInsertInteriorRing
 - gg_core.h, 391
- gaiaInsertIntoSqlLog
 - gaiaaux.h, 130
- gaiaInsertLinestringInGeomColl
 - gg_core.h, 392
- gaiaInsertPolygonInGeomColl
 - gg_core.h, 392
- gaiaInterpolatePoint
 - gg_core.h, 393
- gaiaIntersect
 - gg_core.h, 393
- gaiaIntersectionMatrixPatternMatch
 - gg_advanced.h, 253
- gaiaIntersectionMatrixPatternMatch_r
 - gg_advanced.h, 254
- gaialsClosed
 - gg_advanced.h, 254
- gaialsClosedGeom
 - gg_advanced.h, 255
- gaialsClosedGeom_r
 - gg_advanced.h, 256
- gaialsCompressedXmlBlob
 - gg_xml.h, 570
- gaialsEmpty

gg_core.h, 394
gaialsExifGpsTag
 gaiaexif.h, 152
gaialsGpxXmlBlob
 gg_xml.h, 570
gaialsIsoMetadataXmlBlob
 gg_xml.h, 571
gaialsMapConfigXmlBlob
 gg_xml.h, 571
gaialsNotClosedGeomColl
 gg_core.h, 394
gaialsNotClosedGeomColl_r
 gg_core.h, 395
gaialsNotClosedRing
 gg_core.h, 395
gaialsNotClosedRing_r
 gg_core.h, 397
gaialsPointOnPolygonSurface
 gg_core.h, 397
gaialsPointOnRingSurface
 gg_core.h, 398
gaialsReservedSqliteName
 gaiaaux.h, 130
gaialsReservedSqlName
 gaiaaux.h, 131
gaialsRing
 gg_advanced.h, 256
gaialsRing_r
 gg_advanced.h, 257
gaialsSchemaValidatedXmlBlob
 gg_xml.h, 572
gaialsSimple
 gg_advanced.h, 257
gaialsSimple_r
 gg_advanced.h, 258
gaialsSldSeRasterStyleXmlBlob
 gg_xml.h, 572
gaialsSldSeVectorStyleXmlBlob
 gg_xml.h, 573
gaialsSldStyleXmlBlob
 gg_xml.h, 573
gaialsSvgXmlBlob
 gg_xml.h, 574
gaialsToxic
 gg_core.h, 398
gaialsToxic_r
 gg_core.h, 399
gaialsValid
 gg_advanced.h, 259
gaialsValid_r
 gg_advanced.h, 259
gaialsValidDbfList
 gg_formats.h, 476
gaialsValidDetail
 gg_advanced.h, 260
gaialsValidDetail_r
 gg_advanced.h, 261
gaialsValidDetailEx
 gg_advanced.h, 261
gaialsValidReason
 gg_advanced.h, 263
gaialsValidReason_r
 gg_advanced.h, 263
gaialsValidTrajectory
 gg_core.h, 399
gaialsValidXmlBlob
 gg_xml.h, 574
gaialsValidXPathExpression
 gg_xml.h, 575
gaiaLargestEmptyCircle
 gg_advanced.h, 264
gaiaLargestEmptyCircle_r
 gg_advanced.h, 265
gaiaLayerAttributeFieldInfos, 37
gaiaLayerAttributeFieldPtr
 gg_structs.h, 547
gaiaLayerAuthInfos, 38
gaiaLayerAuthPtr
 gg_structs.h, 547
gaiaLayerExtentInfos, 39
gaiaLayerExtentPtr
 gg_structs.h, 547
gaiaLinearize
 gg_core.h, 400
gaiaLineFromEncodedPolyline
 gg_advanced.h, 265
gaiaLineGetPoint
 gg_core.h, 401
gaiaLineInterpolateEquidistantPoints
 gg_advanced.h, 266
gaiaLineInterpolateEquidistantPoints_r
 gg_advanced.h, 267
gaiaLineInterpolatePoint
 gg_advanced.h, 267
gaiaLineInterpolatePoint_r
 gg_advanced.h, 268
gaiaLineLocatePoint
 gg_advanced.h, 269
gaiaLineLocatePoint_r
 gg_advanced.h, 269
gaiaLineMerge
 gg_advanced.h, 270
gaiaLineMerge_r
 gg_advanced.h, 271
gaiaLinesCutAtNodes
 gg_advanced.h, 271
gaiaLineSetPoint
 gg_core.h, 401
gaiaLinestringEquals
 gg_core.h, 402
gaiaLinestringPtr
 gg_structs.h, 547
gaiaLinestringStruct, 40
gaiaLineSubstring

gg_advanced.h, 272
gaiaLineSubstring_r
gg_advanced.h, 273
gaiaLoadFromDxfParser
gg_dxf.h, 435
gaiaLocateBetweenMeasures
gg_core.h, 403
gaiaMakeArc
gg_core.h, 403
gaiaMakeCircle
gg_core.h, 404
gaiaMakeEllipse
gg_core.h, 405
gaiaMakeEllipticArc
gg_core.h, 405
gaiaMakeLine
gg_formats.h, 477
gaiaMakePoint
gg_formats.h, 477
gaiaMakePointEx
gg_formats.h, 478
gaiaMakePointM
gg_formats.h, 478
gaiaMakePointMEx
gg_formats.h, 479
gaiaMakePointZ
gg_formats.h, 480
gaiaMakePointZEx
gg_formats.h, 480
gaiaMakePointZM
gg_formats.h, 482
gaiaMakePointZMEx
gg_formats.h, 483
gaiaMakePolygon
gg_core.h, 406
gaiaMakeValid
gg_advanced.h, 273
gaiaMakeValidDiscarded
gg_advanced.h, 274
gaiaMaxDistance
gg_advanced.h, 275
gaiaMaximumInscribedCircle
gg_advanced.h, 276
gaiaMaximumInscribedCircle_r
gg_advanced.h, 276
gaiaMbrGeometry
gg_mbr.h, 526
gaiaMbrLinestring
gg_mbr.h, 527
gaiaMbrPolygon
gg_mbr.h, 527
gaiaMbrRing
gg_mbr.h, 527
gaiaMbrsContains
gg_mbr.h, 528
gaiaMbrsDisjoint
gg_mbr.h, 528
gaiaMbrsEqual
gg_mbr.h, 529
gaiaMbrsIntersects
gg_mbr.h, 529
gaiaMbrsOverlaps
gg_mbr.h, 529
gaiaMbrsTouches
gg_mbr.h, 530
gaiaMbrsWithin
gg_mbr.h, 530
gaiaMeasureArea
gg_core.h, 406
gaiaMeasureLength
gg_core.h, 407
gaiaMemFilePtr
gg_structs.h, 548
gaiaMemFileStruct, 41
gaiaMemFseek
gg_formats.h, 483
gaiaMemRead
gg_formats.h, 484
gaiaMergeGeometries
gg_core.h, 407
gaiaMergeGeometries_r
gg_core.h, 408
gaiaMinDistance
gg_core.h, 409
gaiaMinimumBoundingCircle
gg_advanced.h, 277
gaiaMinimumBoundingCircle_r
gg_advanced.h, 278
gaiaMinimumClearance
gg_advanced.h, 278
gaiaMinimumClearance_r
gg_advanced.h, 279
gaiaMinimumClearanceLine
gg_advanced.h, 280
gaiaMinimumClearanceLine_r
gg_advanced.h, 280
gaiaMinimumRotatedRectangle
gg_advanced.h, 281
gaiaMinimumRotatedRectangle_r
gg_advanced.h, 281
gaiaMinimumWidth
gg_advanced.h, 282
gaiaMinimumWidth_r
gg_advanced.h, 283
gaiaMRangeGeometry
gg_mbr.h, 532
gaiaMRangeGeometryEx
gg_mbr.h, 532
gaiaMRangeLinestring
gg_mbr.h, 533
gaiaMRangeLinestringEx
gg_mbr.h, 533
gaiaMRangePolygon
gg_mbr.h, 534
gaiaMRangePolygonEx
gg_mbr.h, 534

gaiaMRangeRing
 gg_mbr.h, 535
gaiaMRangeRingEx
 gg_mbr.h, 535
gaiaNodeLines
 gg_advanced.h, 283
gaiaNormalizeLonLat
 gg_core.h, 409
gaiaOffsetCurve
 gg_advanced.h, 284
gaiaOffsetCurve_r
 gg_advanced.h, 285
gaiaOpenDbfRead
 gg_formats.h, 484
gaiaOpenDbfWrite
 gg_formats.h, 485
gaiaOpenDbfWriteEx
 gg_formats.h, 485
gaiaOpenShpRead
 gg_formats.h, 486
gaiaOpenShpWrite
 gg_formats.h, 487
gaiaOpenShpWriteEx
 gg_formats.h, 487
gaiaOpenZipDbf
 gg_formats.h, 488
gaiaOutBareKml
 gg_formats.h, 489
gaiaOutBufferInitialize
 gg_formats.h, 489
gaiaOutBufferPtr
 gg_structs.h, 548
gaiaOutBufferReset
 gg_formats.h, 490
gaiaOutBufferStruct, 41
gaiaOutFullKml
 gg_formats.h, 490
gaiaOutGeoJSON
 gg_formats.h, 491
gaiaOutGml
 gg_formats.h, 492
gaiaOutGml_ex
 gg_formats.h, 492
gaiaOutLinestringZ
 gg_formats.h, 493
gaiaOutLinestringZex
 gg_formats.h, 493
gaiaOutPointZ
 gg_formats.h, 494
gaiaOutPointZex
 gg_formats.h, 494
gaiaOutPolygonZ
 gg_formats.h, 495
gaiaOutPolygonZex
 gg_formats.h, 495
gaiaOutSvg
 gg_formats.h, 496
gaiaOutWkt
 gg_formats.h, 496
gaiaOutWktEx
 gg_formats.h, 497
gaiaOutWktStrict
 gg_formats.h, 497
gaiaParseDMS
 gaiaaux.h, 131
gaiaParseDxfFile
 gg_dxf.h, 436
gaiaParseDxfFile_r
 gg_dxf.h, 436
gaiaParseEWKT
 gg_formats.h, 498
gaiaParseFilterMbr
 gg_mbr.h, 536
gaiaParseGeoJSON
 gg_formats.h, 498
gaiaParseGml
 gg_formats.h, 499
gaiaParseGml_r
 gg_formats.h, 500
gaiaParseHexEWKB
 gg_formats.h, 500
gaiaParseKml
 gg_formats.h, 501
gaiaParseWkt
 gg_formats.h, 501
gaiaPointPtr
 gg_structs.h, 548
gaiaPointStruct, 42
gaiaPolygonEquals
 gg_core.h, 410
gaiaPolygonize
 gg_advanced.h, 285
gaiaPolygonize_r
 gg_advanced.h, 286
gaiaPolygonPtr
 gg_structs.h, 548
gaiaPolygonStruct, 43
gaiaPrependPointMToDynamicLine
 gg_dynamic.h, 447
gaiaPrependPointToDynamicLine
 gg_dynamic.h, 448
gaiaPrependPointZMToDynamicLine
 gg_dynamic.h, 448
gaiaPrependPointZToDynamicLine
 gg_dynamic.h, 449
gaiaPreRingPtr
 gg_structs.h, 549
gaiaPreRingStruct, 44
gaiaProjAreaPtr
 gg_structs.h, 549
gaiaProjAreaStr, 45
gaiaProjectedPoint
 gg_advanced.h, 287
gaiaQuotedSql
 gaiaaux.h, 132
gaiaRadsToDegs

- gg_advanced.h, 288
- gaiaReadDbfEntity
 - gg_formats.h, 503
- gaiaReadDbfEntity_ex
 - gg_formats.h, 504
- gaiaReadShpEntity
 - gg_formats.h, 504
- gaiaReadShpEntity_ex
 - gg_formats.h, 505
- gaiaReadWktFromZipShp
 - gg_formats.h, 506
- gaiaReducePrecision
 - gg_advanced.h, 288
- gaiaReducePrecision_r
 - gg_advanced.h, 289
- gaiaReflectCoords
 - gg_core.h, 410
- gaiaRemoveExtraSpaces
 - gaiaaux.h, 132
- gaiaRemoveRepeatedPoints
 - gg_core.h, 411
- gaiaRenameColumn
 - spatialite.h, 84
- gaiaRenameTable
 - spatialite.h, 85
- gaiaResetDbfEntity
 - gg_formats.h, 506
- gaiaResetGeosMsg
 - gg_advanced.h, 289
- gaiaResetGeosMsg_r
 - gg_advanced.h, 290
- gaiaResetProjErrorMsg_r
 - gg_advanced.h, 290
- gaiaResetRtTopoMsg
 - gg_advanced.h, 292
- gaiaReverseDynamicLine
 - gg_dynamic.h, 449
- gaiaRingCentroid
 - gg_core.h, 411
- gaiaRingGetPoint
 - gg_core.h, 412
- gaiaRingPtr
 - gg_structs.h, 549
- gaiaRingSetPoint
 - gg_core.h, 413
- gaiaRingStruct, 45
- gaiaRotateCoords
 - gg_core.h, 413
- gaiaSanitize
 - gg_core.h, 414
- gaiaScaleCoords
 - gg_core.h, 414
- gaiaSegmentize
 - gg_advanced.h, 292
- gaiaSetDoubleValue
 - gg_formats.h, 507
- gaiaSetGeosAuxErrorMsg
 - gg_advanced.h, 293
- gaiaSetGeosAuxErrorMsg_r
 - gg_advanced.h, 294
- gaiaSetGeosErrorMsg
 - gg_advanced.h, 294
- gaiaSetGeosErrorMsg_r
 - gg_advanced.h, 295
- gaiaSetGeosWarningMsg
 - gg_advanced.h, 295
- gaiaSetGeosWarningMsg_r
 - gg_advanced.h, 296
- gaiaSetIntValue
 - gg_formats.h, 507
- gaiaSetNullValue
 - gg_formats.h, 508
- gaiaSetPoint
 - gg_const.h, 336
- gaiaSetPointXYM
 - gg_const.h, 336
- gaiaSetPointXYZ
 - gg_const.h, 337
- gaiaSetPointXYZM
 - gg_const.h, 338
- gaiaSetProjDatabasePath
 - gg_advanced.h, 296
- gaiaSetProjErrorMsg_r
 - gg_advanced.h, 297
- gaiaSetRtTopoErrorMsg
 - gg_advanced.h, 297
- gaiaSetRtTopoWarningMsg
 - gg_advanced.h, 298
- gaiaSetStrValue
 - gg_formats.h, 508
- gaiaShapefilePtr
 - gg_structs.h, 549
- gaiaShapefileStruct, 46
- gaiaSharedPaths
 - gg_advanced.h, 298
- gaiaSharedPaths_r
 - gg_advanced.h, 300
- gaiaShiftCoords
 - gg_core.h, 415
- gaiaShiftCoords3D
 - gg_core.h, 415
- gaiaShiftLongitude
 - gg_core.h, 416
- gaiaShortestLine
 - gg_advanced.h, 301
- gaiaShortestLine_r
 - gg_advanced.h, 301
- gaiaShpAnalyze
 - gg_formats.h, 508
- gaiaSingleQuotedSql
 - gaiaaux.h, 133
- gaiaSingleSidedBuffer
 - gg_advanced.h, 302
- gaiaSingleSidedBuffer_r
 - gg_advanced.h, 303
- gaiaSnap

- gg_advanced.h, 303
- gaiaSnap_r
 - gg_advanced.h, 304
- gaiaSnapToGrid
 - gg_advanced.h, 305
- gaiaSplit
 - gg_advanced.h, 306
- gaiaSplitLeft
 - gg_advanced.h, 307
- gaiaSplitRight
 - gg_advanced.h, 307
- gaiaSquareGrid
 - gg_advanced.h, 308
- gaiaSquareGrid_r
 - gg_advanced.h, 309
- gaiaStatisticsInvalidate
 - spatialite.h, 86
- gaiaSubdivide
 - gg_advanced.h, 310
- gaiaSwapCoords
 - gg_core.h, 416
- gaiaTextReaderAlloc
 - gg_formats.h, 509
- gaiaTextReaderDestroy
 - gg_formats.h, 510
- gaiaTextReaderFetchField
 - gg_formats.h, 510
- gaiaTextReaderGetRow
 - gg_formats.h, 510
- gaiaTextReaderParse
 - gg_formats.h, 511
- gaiaTextReaderPtr
 - gg_structs.h, 550
- gaiaToCompressedBlobWkb
 - gg_formats.h, 512
- gaiaToEWKB
 - gg_formats.h, 512
- gaiaToEWKT
 - gg_formats.h, 513
- gaiaToFgf
 - gg_formats.h, 513
- gaiaToGeos
 - gg_advanced.h, 310
- gaiaToGeos_r
 - gg_advanced.h, 311
- gaiaToGeosSelective
 - gg_advanced.h, 312
- gaiaToGeosSelective_r
 - gg_advanced.h, 312
- gaiaToHexWkb
 - gg_formats.h, 514
- gaiaToSpatialLiteBlobWkb
 - gg_formats.h, 514
- gaiaToSpatialLiteBlobWkbEx
 - gg_formats.h, 515
- gaiaToSpatialLiteBlobWkbEx2
 - gg_formats.h, 515
- gaiaToTWKB
 - gg_advanced.h, 313
- gaiaToWkb
 - gg_formats.h, 516
- gaiaTrajectoryInterpolatePoint
 - gg_core.h, 417
- gaiaTransform
 - gg_advanced.h, 314
- gaiaTransform_r
 - gg_advanced.h, 314
- gaiaTransformEx
 - gg_advanced.h, 315
- gaiaTransformEx_r
 - gg_advanced.h, 316
- gaiaTransformXY
 - gg_advanced.h, 317
- gaiaTransformXY_r
 - gg_advanced.h, 317
- gaiaTransformXYZ
 - gg_advanced.h, 318
- gaiaTransformXYZ_r
 - gg_advanced.h, 319
- gaiaTriangularGrid
 - gg_advanced.h, 320
- gaiaTriangularGrid_r
 - gg_advanced.h, 321
- gaiaUnaryUnion
 - gg_advanced.h, 321
- gaiaUnaryUnion_r
 - gg_advanced.h, 322
- gaiaUnionCascaded
 - gg_advanced.h, 323
- gaiaUnionCascaded_r
 - gg_advanced.h, 323
- gaiaUpdateMD5Checksum
 - gaiaaux.h, 133
- gaiaUpdateMetaCatalogStatistics
 - spatialite.h, 86
- gaiaUpdateMetaCatalogStatisticsFromMaster
 - spatialite.h, 87
- gaiaUpdateSqlLog
 - gaiaaux.h, 134
- gaiaUpDownHeight
 - gg_core.h, 417
- gaiaValueStruct, 48
- gaiaVectorLayerItem, 49
- gaiaVectorLayerPtr
 - gg_structs.h, 550
- gaiaVectorLayersListPtr
 - gg_structs.h, 550
- gaiaVectorLayersListStr, 50
- gaiaVoronoiDiagram
 - gg_advanced.h, 324
- gaiaVoronoiDiagram_r
 - gg_advanced.h, 325
- gaiaWriteDbfEntity
 - gg_formats.h, 516
- gaiaWriteShpEntity
 - gg_formats.h, 517

gaiaXmlBlobAddFileId
gg_xml.h, 575

gaiaXmlBlobAddParentId
gg_xml.h, 576

gaiaXmlBlobCompression
gg_xml.h, 577

gaiaXmlBlobGetAbstract
gg_xml.h, 578

gaiaXmlBlobGetDocumentSize
gg_xml.h, 578

gaiaXmlBlobGetEncoding
gg_xml.h, 579

gaiaXmlBlobGetFileId
gg_xml.h, 579

gaiaXmlBlobGetGeometry
gg_xml.h, 580

gaiaXmlBlobGetLastParseError
gg_xml.h, 580

gaiaXmlBlobGetLastValidateError
gg_xml.h, 581

gaiaXmlBlobGetLastXPathError
gg_xml.h, 581

gaiaXmlBlobGetName
gg_xml.h, 582

gaiaXmlBlobGetParentId
gg_xml.h, 582

gaiaXmlBlobGetSchemaURI
gg_xml.h, 583

gaiaXmlBlobGetTitle
gg_xml.h, 584

gaiaXmlBlobMLineFromGPX
gg_xml.h, 584

gaiaXmlBlobSetFileId
gg_xml.h, 585

gaiaXmlBlobSetParentId
gg_xml.h, 585

gaiaXmlFromBlob
gg_xml.h, 586

gaiaXmlGetInternalSchemaURI
gg_xml.h, 587

gaiaXmlLoad
gg_xml.h, 587

gaiaXmlStore
gg_xml.h, 588

gaiaXmlTextFromBlob
gg_xml.h, 589

gaiaXmlToBlob
gg_xml.h, 589

gaiaZipfileDbfN
gg_formats.h, 518

gaiaZipfileNumDBF
gg_formats.h, 518

gaiaZipfileNumSHP
gg_formats.h, 518

gaiaZipfileShpN
gg_formats.h, 519

gaiaZRangeGeometry
gg_mbr.h, 536

gaiaZRangeGeometryEx
gg_mbr.h, 537

gaiaZRangeLinestring
gg_mbr.h, 537

gaiaZRangeLinestringEx
gg_mbr.h, 538

gaiaZRangePolygon
gg_mbr.h, 538

gaiaZRangePolygonEx
gg_mbr.h, 539

gaiaZRangeRing
gg_mbr.h, 539

gaiaZRangeRingEx
gg_mbr.h, 540

get_wfs_base_describe_url
gg_wfs.h, 554

get_wfs_base_request_url
gg_wfs.h, 554

get_wfs_catalog_count
gg_wfs.h, 555

get_wfs_catalog_item
gg_wfs.h, 555

get_wfs_describe_url
gg_wfs.h, 556

get_wfs_item_abstract
gg_wfs.h, 556

get_wfs_item_name
gg_wfs.h, 557

get_wfs_item_title
gg_wfs.h, 557

get_wfs_keyword
gg_wfs.h, 558

get_wfs_keyword_count
gg_wfs.h, 558

get_wfs_layer_srid
gg_wfs.h, 559

get_wfs_layer_srid_count
gg_wfs.h, 559

get_wfs_request_url
gg_wfs.h, 560

get_wfs_schema_column
gg_wfs.h, 560

get_wfs_schema_column_count
gg_wfs.h, 561

get_wfs_schema_column_info
gg_wfs.h, 561

get_wfs_schema_geometry_info
gg_wfs.h, 562

get_wfs_version
gg_wfs.h, 562

gg_advanced.h
gaia3DDistance, 166
gaia3dLength, 167
gaia3DMaxDistance, 168
gaiaAsEncodedPolyLine, 168
gaiaAsX3D, 169
gaiaAzimuth, 170
gaiaBoundary, 170

- gaiaBoundary_r, 171
- gaiaConcaveHull, 172
- gaiaConcaveHull_r, 173
- gaiaConstrainedDelaunayTriangulation, 173
- gaiaConstrainedDelaunayTriangulation_r, 174
- gaiaConvexHull, 175
- gaiaConvexHull_r, 175
- gaiaCriticalPointFromGEOSmsg, 176
- gaiaCriticalPointFromGEOSmsg_r, 176
- gaiaDegsToRads, 177
- gaiaDelaunayTriangulation, 177
- gaiaDelaunayTriangulation_r, 178
- gaiaDrapeLine, 179
- gaiaDrapeLineExceptions, 180
- gaiaEllipsoidAzimuth, 180
- gaiaFrechetDistance, 181
- gaiaFrechetDistance_r, 182
- gaiaFrechetDistanceDensify, 182
- gaiaFrechetDistanceDensify_r, 183
- gaiaFromGeos_XY, 184
- gaiaFromGeos_XY_r, 185
- gaiaFromGeos_XYM, 185
- gaiaFromGeos_XYM_r, 186
- gaiaFromGeos_XYZ, 187
- gaiaFromGeos_XYZ_r, 187
- gaiaFromGeos_XYZM, 188
- gaiaFromGeos_XYZM_r, 189
- gaiaFromTWKB, 189
- gaiaGeodesicArea, 190
- gaiaGeoHash, 191
- gaiaGeomCollArea, 191
- gaiaGeomCollArea_r, 192
- gaiaGeomCollBuffer, 193
- gaiaGeomCollBuffer_r, 193
- gaiaGeomCollCentroid, 194
- gaiaGeomCollCentroid_r, 195
- gaiaGeomCollContains, 195
- gaiaGeomCollContains_r, 196
- gaiaGeomCollCoveredBy, 197
- gaiaGeomCollCoveredBy_r, 197
- gaiaGeomCollCovers, 198
- gaiaGeomCollCovers_r, 199
- gaiaGeomCollCrosses, 199
- gaiaGeomCollCrosses_r, 200
- gaiaGeomCollDisjoint, 201
- gaiaGeomCollDisjoint_r, 201
- gaiaGeomCollDistance, 202
- gaiaGeomCollDistance_r, 203
- gaiaGeomCollEquals, 204
- gaiaGeomCollEquals_r, 204
- gaiaGeomCollIntersects, 205
- gaiaGeomCollIntersects_r, 206
- gaiaGeomCollLength, 206
- gaiaGeomCollLength_r, 207
- gaiaGeomCollLengthOrPerimeter, 208
- gaiaGeomCollLengthOrPerimeter_r, 208
- gaiaGeomCollOverlaps, 209
- gaiaGeomCollOverlaps_r, 210
- gaiaGeomCollPreparedContains, 210
- gaiaGeomCollPreparedCoveredBy, 211
- gaiaGeomCollPreparedCovers, 212
- gaiaGeomCollPreparedCrosses, 213
- gaiaGeomCollPreparedDisjoint, 214
- gaiaGeomCollPreparedDistance, 214
- gaiaGeomCollPreparedDistanceWithin, 215
- gaiaGeomCollPreparedIntersects, 216
- gaiaGeomCollPreparedOverlaps, 217
- gaiaGeomCollPreparedTouches, 218
- gaiaGeomCollPreparedWithin, 218
- gaiaGeomCollRelate, 219
- gaiaGeomCollRelate_r, 220
- gaiaGeomCollRelateBoundaryNodeRule, 221
- gaiaGeomCollRelateBoundaryNodeRule_r, 221
- gaiaGeomCollSimplify, 222
- gaiaGeomCollSimplify_r, 223
- gaiaGeomCollSimplifyPreserveTopology, 224
- gaiaGeomCollSimplifyPreserveTopology_r, 224
- gaiaGeomCollTouches, 225
- gaiaGeomCollTouches_r, 226
- gaiaGeomCollWithin, 226
- gaiaGeomCollWithin_r, 227
- gaiaGeometryDifference, 228
- gaiaGeometryDifference_r, 228
- gaiaGeometryIntersection, 229
- gaiaGeometryIntersection_r, 230
- gaiaGeometrySymDifference, 230
- gaiaGeometrySymDifference_r, 232
- gaiaGeometryUnion, 233
- gaiaGeometryUnion_r, 233
- gaiaGeosConcaveHull, 234
- gaiaGeosConcaveHull_r, 235
- gaiaGeosDensify, 236
- gaiaGeosDensify_r, 236
- gaiaGeosMakeValid, 237
- gaiaGeosMakeValid_r, 238
- gaiaGetGeosAuxErrorMsg, 238
- gaiaGetGeosAuxErrorMsg_r, 239
- gaiaGetGeosErrorMsg, 240
- gaiaGetGeosErrorMsg_r, 240
- gaiaGetGeosWarningMsg, 241
- gaiaGetGeosWarningMsg_r, 241
- gaiaGetPointOnSurface, 242
- gaiaGetPointOnSurface_r, 243
- gaiaGetProjDatabasePath, 243
- gaiaGetProjErrorMsg_r, 244
- gaiaGetProjString, 244
- gaiaGetProjWKT, 245
- gaiaGetRtTopoErrorMsg, 246
- gaiaGetRtTopoWarningMsg, 246
- gaiaGuessSridFromWKT, 247
- gaiaHausdorffDistance, 247
- gaiaHausdorffDistance_r, 248
- gaiaHausdorffDistanceDensify, 249
- gaiaHausdorffDistanceDensify_r, 249
- gaiaHexagonalGrid, 250
- gaiaHexagonalGrid_r, 251

gaiaHilbertCode, 252
 gaiaHilbertCode_r, 252
 gaiaIntersectionMatrixPatternMatch, 253
 gaiaIntersectionMatrixPatternMatch_r, 254
 gaialsClosed, 254
 gaialsClosedGeom, 255
 gaialsClosedGeom_r, 256
 gaialsRing, 256
 gaialsRing_r, 257
 gaialsSimple, 257
 gaialsSimple_r, 258
 gaialsValid, 259
 gaialsValid_r, 259
 gaialsValidDetail, 260
 gaialsValidDetail_r, 261
 gaialsValidDetailEx, 261
 gaialsValidDetailEx_r, 262
 gaialsValidReason, 263
 gaialsValidReason_r, 263
 gaiaLargestEmptyCircle, 264
 gaiaLargestEmptyCircle_r, 265
 gaiaLineFromEncodedPolyline, 265
 gaiaLineInterpolateEquidistantPoints, 266
 gaiaLineInterpolateEquidistantPoints_r, 267
 gaiaLineInterpolatePoint, 267
 gaiaLineInterpolatePoint_r, 268
 gaiaLineLocatePoint, 269
 gaiaLineLocatePoint_r, 269
 gaiaLineMerge, 270
 gaiaLineMerge_r, 271
 gaiaLinesCutAtNodes, 271
 gaiaLineSubstring, 272
 gaiaLineSubstring_r, 273
 gaiaMakeValid, 273
 gaiaMakeValidDiscarded, 274
 gaiaMaxDistance, 275
 gaiaMaximumInscribedCircle, 276
 gaiaMaximumInscribedCircle_r, 276
 gaiaMinimumBoundingCircle, 277
 gaiaMinimumBoundingCircle_r, 278
 gaiaMinimumClearance, 278
 gaiaMinimumClearance_r, 279
 gaiaMinimumClearanceLine, 280
 gaiaMinimumClearanceLine_r, 280
 gaiaMinimumRotatedRectangle, 281
 gaiaMinimumRotatedRectangle_r, 281
 gaiaMinimumWidth, 282
 gaiaMinimumWidth_r, 283
 gaiaNodeLines, 283
 gaiaOffsetCurve, 284
 gaiaOffsetCurve_r, 285
 gaiaPolygonize, 285
 gaiaPolygonize_r, 286
 gaiaProjectedPoint, 287
 gaiaRadsToDegs, 288
 gaiaReducePrecision, 288
 gaiaReducePrecision_r, 289
 gaiaResetGeosMsg, 289
 gaiaResetGeosMsg_r, 290
 gaiaResetProjErrorMsg_r, 290
 gaiaResetRtTopoMsg, 292
 gaiaSegmentize, 292
 gaiaSetGeosAuxErrorMsg, 293
 gaiaSetGeosAuxErrorMsg_r, 294
 gaiaSetGeosErrorMsg, 294
 gaiaSetGeosErrorMsg_r, 295
 gaiaSetGeosWarningMsg, 295
 gaiaSetGeosWarningMsg_r, 296
 gaiaSetProjDatabasePath, 296
 gaiaSetProjErrorMsg_r, 297
 gaiaSetRtTopoErrorMsg, 297
 gaiaSetRtTopoWarningMsg, 298
 gaiaSharedPaths, 298
 gaiaSharedPaths_r, 300
 gaiaShortestLine, 301
 gaiaShortestLine_r, 301
 gaiaSingleSidedBuffer, 302
 gaiaSingleSidedBuffer_r, 303
 gaiaSnap, 303
 gaiaSnap_r, 304
 gaiaSnapToGrid, 305
 gaiaSplit, 306
 gaiaSplitLeft, 307
 gaiaSplitRight, 307
 gaiaSquareGrid, 308
 gaiaSquareGrid_r, 309
 gaiaSubdivide, 310
 gaiaToGeos, 310
 gaiaToGeos_r, 311
 gaiaToGeosSelective, 312
 gaiaToGeosSelective_r, 312
 gaiaToTWKB, 313
 gaiaTransform, 314
 gaiaTransform_r, 314
 gaiaTransformEx, 315
 gaiaTransformEx_r, 316
 gaiaTransformXY, 317
 gaiaTransformXY_r, 317
 gaiaTransformXYZ, 318
 gaiaTransformXYZ_r, 319
 gaiaTriangularGrid, 320
 gaiaTriangularGrid_r, 321
 gaiaUnaryUnion, 321
 gaiaUnaryUnion_r, 322
 gaiaUnionCascaded, 323
 gaiaUnionCascaded_r, 323
 gaiaVoronoiDiagram, 324
 gaiaVoronoiDiagram_r, 325
 gg_const.h
 gaiaGetPoint, 333
 gaiaGetPointXYM, 334
 gaiaGetPointXYZ, 334
 gaiaGetPointXYZM, 335
 gaiaSetPoint, 336
 gaiaSetPointXYM, 336
 gaiaSetPointXYZ, 337

- gaiaSetPointXYZM, [338](#)
- gg_core.h
 - gaiaAddInteriorRing, [345](#)
 - gaiaAddLinestringToGeomColl, [346](#)
 - gaiaAddMeasure, [346](#)
 - gaiaAddPointToGeomColl, [347](#)
 - gaiaAddPointToGeomCollXYM, [347](#)
 - gaiaAddPointToGeomCollXYZ, [348](#)
 - gaiaAddPointToGeomCollXYZM, [348](#)
 - gaiaAddPolygonToGeomColl, [349](#)
 - gaiaAddRingToPolyg, [349](#)
 - gaiaAllocGeomColl, [350](#)
 - gaiaAllocGeomCollXYM, [350](#)
 - gaiaAllocGeomCollXYZ, [351](#)
 - gaiaAllocGeomCollXYZM, [351](#)
 - gaiaAllocLinestring, [352](#)
 - gaiaAllocLinestringXYM, [352](#)
 - gaiaAllocLinestringXYZ, [353](#)
 - gaiaAllocLinestringXYZM, [354](#)
 - gaiaAllocPoint, [354](#)
 - gaiaAllocPointXYM, [355](#)
 - gaiaAllocPointXYZ, [355](#)
 - gaiaAllocPointXYZM, [357](#)
 - gaiaAllocPolygon, [358](#)
 - gaiaAllocPolygonXYM, [358](#)
 - gaiaAllocPolygonXYZ, [359](#)
 - gaiaAllocPolygonXYZM, [359](#)
 - gaiaAllocRing, [360](#)
 - gaiaAllocRingXYM, [360](#)
 - gaiaAllocRingXYZ, [361](#)
 - gaiaAllocRingXYZM, [362](#)
 - gaiaCastGeomCollToXY, [362](#)
 - gaiaCastGeomCollToXYM, [363](#)
 - gaiaCastGeomCollToXYMnoData, [363](#)
 - gaiaCastGeomCollToXYZ, [364](#)
 - gaiaCastGeomCollToXYZM, [364](#)
 - gaiaCastGeomCollToXYZMnoData, [366](#)
 - gaiaCastGeomCollToXYZnoData, [366](#)
 - gaiaCheckClockwise, [367](#)
 - gaiaCheckCounterClockwise, [367](#)
 - gaiaClockwise, [368](#)
 - gaiaCloneGeomColl, [368](#)
 - gaiaCloneGeomCollLinestrings, [369](#)
 - gaiaCloneGeomCollPoints, [369](#)
 - gaiaCloneGeomCollPolygons, [370](#)
 - gaiaCloneGeomCollSpecial, [370](#)
 - gaiaCloneLinestring, [371](#)
 - gaiaCloneLinestringSpecial, [372](#)
 - gaiaClonePolygon, [372](#)
 - gaiaClonePolygonSpecial, [373](#)
 - gaiaCloneRing, [373](#)
 - gaiaCloneRingSpecial, [374](#)
 - gaiaConvertLength, [374](#)
 - gaiaCopyLinestringCoords, [375](#)
 - gaiaCopyLinestringCoordsEx, [375](#)
 - gaiaCopyLinestringCoordsReverse, [376](#)
 - gaiaCopyRingCoords, [376](#)
 - gaiaCopyRingCoordsEx, [377](#)
 - gaiaCopyRingCoordsReverse, [377](#)
 - gaiaCreatePolygon, [378](#)
 - gaiaCurvosityIndex, [378](#)
 - gaiaDimension, [379](#)
 - gaiaDissolvePoints, [379](#)
 - gaiaDissolveSegments, [380](#)
 - gaiaEllipseParams, [381](#)
 - gaiaEnsureClosedRings, [381](#)
 - gaiaExtractLinestringsFromGeomColl, [382](#)
 - gaiaExtractPointsFromGeomColl, [382](#)
 - gaiaExtractPolygonsFromGeomColl, [383](#)
 - gaiaFree, [384](#)
 - gaiaFreeGeomColl, [384](#)
 - gaiaFreeLinestring, [385](#)
 - gaiaFreePoint, [385](#)
 - gaiaFreePolygon, [385](#)
 - gaiaFreeRing, [386](#)
 - gaiaGeodesicArcLength, [386](#)
 - gaiaGeodesicDistance, [387](#)
 - gaiaGeodesicTotalLength, [388](#)
 - gaiaGeometryAliasType, [389](#)
 - gaiaGeometryType, [389](#)
 - gaiaGreatCircleDistance, [390](#)
 - gaiaGreatCircleTotalLength, [391](#)
 - gaiaInsertInteriorRing, [391](#)
 - gaiaInsertLinestringInGeomColl, [392](#)
 - gaiaInsertPolygonInGeomColl, [392](#)
 - gaiaInterpolatePoint, [393](#)
 - gaiaIntersect, [393](#)
 - gaialsEmpty, [394](#)
 - gaialsNotClosedGeomColl, [394](#)
 - gaialsNotClosedGeomColl_r, [395](#)
 - gaialsNotClosedRing, [395](#)
 - gaialsNotClosedRing_r, [397](#)
 - gaialsPointOnPolygonSurface, [397](#)
 - gaialsPointOnRingSurface, [398](#)
 - gaialsToxic, [398](#)
 - gaialsToxic_r, [399](#)
 - gaialsValidTrajectory, [399](#)
 - gaiaLinearize, [400](#)
 - gaiaLineGetPoint, [401](#)
 - gaiaLineSetPoint, [401](#)
 - gaiaLinestringEquals, [402](#)
 - gaiaLocateBetweenMeasures, [403](#)
 - gaiaMakeArc, [403](#)
 - gaiaMakeCircle, [404](#)
 - gaiaMakeEllipse, [405](#)
 - gaiaMakeEllipticArc, [405](#)
 - gaiaMakePolygon, [406](#)
 - gaiaMeasureArea, [406](#)
 - gaiaMeasureLength, [407](#)
 - gaiaMergeGeometries, [407](#)
 - gaiaMergeGeometries_r, [408](#)
 - gaiaMinDistance, [409](#)
 - gaiaNormalizeLonLat, [409](#)
 - gaiaPolygonEquals, [410](#)
 - gaiaReflectCoords, [410](#)
 - gaiaRemoveRepeatedPoints, [411](#)

- gaiaRingCentroid, [411](#)
- gaiaRingGetPoint, [412](#)
- gaiaRingSetPoint, [413](#)
- gaiaRotateCoords, [413](#)
- gaiaSanitize, [414](#)
- gaiaScaleCoords, [414](#)
- gaiaShiftCoords, [415](#)
- gaiaShiftCoords3D, [415](#)
- gaiaShiftLongitude, [416](#)
- gaiaSwapCoords, [416](#)
- gaiaTrajectoryInterpolatePoint, [417](#)
- gaiaUpDownHeight, [417](#)
- gg_dxf.h
 - gaiaCreateDxfParser, [426](#)
 - gaiaDestroyDxfParser, [426](#)
 - gaiaDxfArcPtr, [422](#)
 - gaiaDxfBlockPtr, [422](#)
 - gaiaDxfBoundaryPathPtr, [423](#)
 - gaiaDxfCirclePtr, [423](#)
 - gaiaDxfExtraAttrPtr, [423](#)
 - gaiaDxfHatchPtr, [423](#)
 - gaiaDxfHatchSegmPtr, [424](#)
 - gaiaDxfHolePtr, [424](#)
 - gaiaDxfInsertPtr, [424](#)
 - gaiaDxfLayerPtr, [424](#)
 - gaiaDxfParserPtr, [425](#)
 - gaiaDxfPointPtr, [425](#)
 - gaiaDxfPolylinePtr, [425](#)
 - gaiaDxfTextPtr, [425](#)
 - gaiaDxfWriteEndSection, [427](#)
 - gaiaDxfWriteEntities, [427](#)
 - gaiaDxfWriteFooter, [428](#)
 - gaiaDxfWriteGeometry, [428](#)
 - gaiaDxfWriteHeader, [429](#)
 - gaiaDxfWriteLayer, [429](#)
 - gaiaDxfWriteLine, [430](#)
 - gaiaDxfWritePoint, [430](#)
 - gaiaDxfWriteRing, [431](#)
 - gaiaDxfWriterInit, [432](#)
 - gaiaDxfWriteTables, [432](#)
 - gaiaDxfWriteText, [433](#)
 - gaiaExportDxf, [433](#)
 - gaiaLoadFromDxfParser, [435](#)
 - gaiaParseDxfFile, [436](#)
 - gaiaParseDxfFile_r, [436](#)
- gg_dynamic.h
 - gaiaAllocDynamicLine, [439](#)
 - gaiaAppendPointMToDynamicLine, [439](#)
 - gaiaAppendPointToDynamicLine, [440](#)
 - gaiaAppendPointZMToDynamicLine, [440](#)
 - gaiaAppendPointZToDynamicLine, [440](#)
 - gaiaCloneDynamicLine, [441](#)
 - gaiaCreateDynamicLine, [441](#)
 - gaiaDynamicLineDeletePoint, [442](#)
 - gaiaDynamicLineFindByCoords, [442](#)
 - gaiaDynamicLineFindByPos, [443](#)
 - gaiaDynamicLineInsertAfter, [443](#)
 - gaiaDynamicLineInsertBefore, [444](#)
 - gaiaDynamicLineJoinAfter, [444](#)
 - gaiaDynamicLineJoinBefore, [445](#)
 - gaiaDynamicLineSplitAfter, [446](#)
 - gaiaDynamicLineSplitBefore, [446](#)
 - gaiaFreeDynamicLine, [447](#)
 - gaiaPrependPointMToDynamicLine, [447](#)
 - gaiaPrependPointToDynamicLine, [448](#)
 - gaiaPrependPointZMToDynamicLine, [448](#)
 - gaiaPrependPointZToDynamicLine, [449](#)
 - gaiaReverseDynamicLine, [449](#)
- gg_formats.h
 - gaiaAddDbfField, [455](#)
 - gaiaAllocDbf, [456](#)
 - gaiaAllocDbfField, [457](#)
 - gaiaAllocDbfList, [457](#)
 - gaiaAllocShapefile, [458](#)
 - gaiaAppendToOutBuffer, [458](#)
 - gaiaCloneDbfEntity, [459](#)
 - gaiaCloneDbfField, [459](#)
 - gaiaCloneValue, [460](#)
 - gaiaEndianArch, [460](#)
 - gaiaEwkbGetLinestring, [461](#)
 - gaiaEwkbGetMultiGeometry, [461](#)
 - gaiaEwkbGetPoint, [462](#)
 - gaiaEwkbGetPolygon, [463](#)
 - gaiaExport16, [464](#)
 - gaiaExport32, [464](#)
 - gaiaExport64, [465](#)
 - gaiaExportF32, [465](#)
 - gaiaExportI64, [466](#)
 - gaiaExportU32, [466](#)
 - gaiaFlushDbfHeader, [467](#)
 - gaiaFlushShpHeaders, [467](#)
 - gaiaFreeDbf, [468](#)
 - gaiaFreeDbfField, [468](#)
 - gaiaFreeDbfList, [468](#)
 - gaiaFreeShapefile, [469](#)
 - gaiaFreeValue, [469](#)
 - gaiaFromEWKB, [470](#)
 - gaiaFromFgf, [470](#)
 - gaiaFromSpatialLiteBlobWkb, [471](#)
 - gaiaFromSpatialLiteBlobWkbEx, [471](#)
 - gaiaFromWkb, [472](#)
 - gaiaImport16, [473](#)
 - gaiaImport32, [473](#)
 - gaiaImport64, [474](#)
 - gaiaImportF32, [474](#)
 - gaiaImportI64, [475](#)
 - gaiaImportU32, [476](#)
 - gaiaIsValidDbfList, [476](#)
 - gaiaMakeLine, [477](#)
 - gaiaMakePoint, [477](#)
 - gaiaMakePointEx, [478](#)
 - gaiaMakePointM, [478](#)
 - gaiaMakePointMEx, [479](#)
 - gaiaMakePointZ, [480](#)
 - gaiaMakePointZEx, [480](#)
 - gaiaMakePointZM, [482](#)

- gaiaMakePointZMEx, [483](#)
- gaiaMemFseek, [483](#)
- gaiaMemRead, [484](#)
- gaiaOpenDbfRead, [484](#)
- gaiaOpenDbfWrite, [485](#)
- gaiaOpenDbfWriteEx, [485](#)
- gaiaOpenShpRead, [486](#)
- gaiaOpenShpWrite, [487](#)
- gaiaOpenShpWriteEx, [487](#)
- gaiaOpenZipDbf, [488](#)
- gaiaOutBareKml, [489](#)
- gaiaOutBufferInitialize, [489](#)
- gaiaOutBufferReset, [490](#)
- gaiaOutFullKml, [490](#)
- gaiaOutGeoJSON, [491](#)
- gaiaOutGml, [492](#)
- gaiaOutGml_ex, [492](#)
- gaiaOutLinestringZ, [493](#)
- gaiaOutLinestringZex, [493](#)
- gaiaOutPointZ, [494](#)
- gaiaOutPointZex, [494](#)
- gaiaOutPolygonZ, [495](#)
- gaiaOutPolygonZex, [495](#)
- gaiaOutSvg, [496](#)
- gaiaOutWkt, [496](#)
- gaiaOutWktEx, [497](#)
- gaiaOutWktStrict, [497](#)
- gaiaParseEWKT, [498](#)
- gaiaParseGeoJSON, [498](#)
- gaiaParseGml, [499](#)
- gaiaParseGml_r, [500](#)
- gaiaParseHexEWKB, [500](#)
- gaiaParseKml, [501](#)
- gaiaParseWkt, [501](#)
- gaiaReadDbfEntity, [503](#)
- gaiaReadDbfEntity_ex, [504](#)
- gaiaReadShpEntity, [504](#)
- gaiaReadShpEntity_ex, [505](#)
- gaiaReadWktFromZipShp, [506](#)
- gaiaResetDbfEntity, [506](#)
- gaiaSetDoubleValue, [507](#)
- gaiaSetIntValue, [507](#)
- gaiaSetNullValue, [508](#)
- gaiaSetStrValue, [508](#)
- gaiaShpAnalyze, [508](#)
- gaiaTextReaderAlloc, [509](#)
- gaiaTextReaderDestroy, [510](#)
- gaiaTextReaderFetchField, [510](#)
- gaiaTextReaderGetRow, [510](#)
- gaiaTextReaderParse, [511](#)
- gaiaToCompressedBlobWkb, [512](#)
- gaiaToEWKB, [512](#)
- gaiaToEWKT, [513](#)
- gaiaToFgf, [513](#)
- gaiaToHexWkb, [514](#)
- gaiaToSpatialLiteBlobWkb, [514](#)
- gaiaToSpatialLiteBlobWkbEx, [515](#)
- gaiaToSpatialLiteBlobWkbEx2, [515](#)
- gaiaToWkb, [516](#)
- gaiaWriteDbfEntity, [516](#)
- gaiaWriteShpEntity, [517](#)
- gaiaZipfileDbfN, [518](#)
- gaiaZipfileNumDBF, [518](#)
- gaiaZipfileNumSHP, [518](#)
- gaiaZipfileShpN, [519](#)
- gg_mbr.h
 - gaiaBuildCircleMbr, [522](#)
 - gaiaBuildFilterMbr, [522](#)
 - gaiaBuildMbr, [523](#)
 - gaiaFromSpatialLiteBlobMbr, [524](#)
 - gaiaGetMbrMaxX, [524](#)
 - gaiaGetMbrMaxY, [525](#)
 - gaiaGetMbrMinX, [525](#)
 - gaiaGetMbrMinY, [526](#)
 - gaiaMbrGeometry, [526](#)
 - gaiaMbrLinestring, [527](#)
 - gaiaMbrPolygon, [527](#)
 - gaiaMbrRing, [527](#)
 - gaiaMbrsContains, [528](#)
 - gaiaMbrsDisjoint, [528](#)
 - gaiaMbrsEqual, [529](#)
 - gaiaMbrsIntersects, [529](#)
 - gaiaMbrsOverlaps, [529](#)
 - gaiaMbrsTouches, [530](#)
 - gaiaMbrsWithin, [530](#)
 - gaiaMRangeGeometry, [532](#)
 - gaiaMRangeGeometryEx, [532](#)
 - gaiaMRangeLinestring, [533](#)
 - gaiaMRangeLinestringEx, [533](#)
 - gaiaMRangePolygon, [534](#)
 - gaiaMRangePolygonEx, [534](#)
 - gaiaMRangeRing, [535](#)
 - gaiaMRangeRingEx, [535](#)
 - gaiaParseFilterMbr, [536](#)
 - gaiaZRangeGeometry, [536](#)
 - gaiaZRangeGeometryEx, [537](#)
 - gaiaZRangeLinestring, [537](#)
 - gaiaZRangeLinestringEx, [538](#)
 - gaiaZRangePolygon, [538](#)
 - gaiaZRangePolygonEx, [539](#)
 - gaiaZRangeRing, [539](#)
 - gaiaZRangeRingEx, [540](#)
- gg_structs.h
 - gaiaAttributeFieldDoubleRangePtr, [545](#)
 - gaiaAttributeFieldIntRangePtr, [545](#)
 - gaiaAttributeFieldMaxSizePtr, [545](#)
 - gaiaDbfListPtr, [546](#)
 - gaiaDbfPtr, [546](#)
 - gaiaDynamicLinePtr, [546](#)
 - gaiaGeomCollPtr, [546](#)
 - gaiaLayerAttributeFieldPtr, [547](#)
 - gaiaLayerAuthPtr, [547](#)
 - gaiaLayerExtentPtr, [547](#)
 - gaiaLinestringPtr, [547](#)
 - gaiaMemFilePtr, [548](#)
 - gaiaOutBufferPtr, [548](#)

- gaiaPointPtr, 548
- gaiaPolygonPtr, 548
- gaiaPreRingPtr, 549
- gaiaProjAreaPtr, 549
- gaiaRingPtr, 549
- gaiaShapefilePtr, 549
- gaiaTextReaderPtr, 550
- gaiaVectorLayerPtr, 550
- gaiaVectorLayersListPtr, 550
- gg_wfs.h
 - create_wfs_catalog, 552
 - create_wfs_schema, 553
 - destroy_wfs_catalog, 553
 - destroy_wfs_schema, 554
 - get_wfs_base_describe_url, 554
 - get_wfs_base_request_url, 554
 - get_wfs_catalog_count, 555
 - get_wfs_catalog_item, 555
 - get_wfs_describe_url, 556
 - get_wfs_item_abstract, 556
 - get_wfs_item_name, 557
 - get_wfs_item_title, 557
 - get_wfs_keyword, 558
 - get_wfs_keyword_count, 558
 - get_wfs_layer_srid, 559
 - get_wfs_layer_srid_count, 559
 - get_wfs_request_url, 560
 - get_wfs_schema_column, 560
 - get_wfs_schema_column_count, 561
 - get_wfs_schema_column_info, 561
 - get_wfs_schema_geometry_info, 562
 - get_wfs_version, 562
 - load_from_wfs, 563
 - load_from_wfs_paged, 564
 - load_from_wfs_paged_ex, 565
 - reset_wfs_http_connection, 566
- gg_xml.h
 - gaia_libxml2_version, 569
 - gaialsCompressedXmlBlob, 570
 - gaialsGpxXmlBlob, 570
 - gaialsIsoMetadataXmlBlob, 571
 - gaialsMapConfigXmlBlob, 571
 - gaialsSchemaValidatedXmlBlob, 572
 - gaialsSldSeRasterStyleXmlBlob, 572
 - gaialsSldSeVectorStyleXmlBlob, 573
 - gaialsSldStyleXmlBlob, 573
 - gaialsSvgXmlBlob, 574
 - gaialsValidXmlBlob, 574
 - gaialsValidXPathExpression, 575
 - gaiaXmlBlobAddFileId, 575
 - gaiaXmlBlobAddParentId, 576
 - gaiaXmlBlobCompression, 577
 - gaiaXmlBlobGetAbstract, 578
 - gaiaXmlBlobGetDocumentSize, 578
 - gaiaXmlBlobGetEncoding, 579
 - gaiaXmlBlobGetFileId, 579
 - gaiaXmlBlobGetGeometry, 580
 - gaiaXmlBlobGetLastError, 580
 - gaiaXmlBlobGetLastValidateError, 581
 - gaiaXmlBlobGetLastXPathError, 581
 - gaiaXmlBlobGetName, 582
 - gaiaXmlBlobGetParentId, 582
 - gaiaXmlBlobGetSchemaURI, 583
 - gaiaXmlBlobGetTitle, 584
 - gaiaXmlBlobMLineFromGPX, 584
 - gaiaXmlBlobSetFileId, 585
 - gaiaXmlBlobSetParentId, 585
 - gaiaXmlFromBlob, 586
 - gaiaXmlGetInternalSchemaURI, 587
 - gaiaXmlLoad, 587
 - gaiaXmlStore, 588
 - gaiaXmlTextFromBlob, 589
 - gaiaXmlToBlob, 589
- insert_epsg_srid
 - spatialite.h, 87
- is_kml_constant
 - spatialite.h, 88
- is_tiny_point_enabled
 - spatialite.h, 88
- load_dbf
 - spatialite.h, 89
- load_dbf_ex
 - spatialite.h, 89
- load_dbf_ex2
 - spatialite.h, 90
- load_dbf_ex3
 - spatialite.h, 91
- load_from_wfs
 - gg_wfs.h, 563
- load_from_wfs_paged
 - gg_wfs.h, 564
- load_from_wfs_paged_ex
 - gg_wfs.h, 565
- load_geojson
 - spatialite.h, 92
- load_shapefile
 - spatialite.h, 92
- load_shapefile_ex
 - spatialite.h, 93
- load_shapefile_ex2
 - spatialite.h, 95
- load_shapefile_ex3
 - spatialite.h, 96
- load_XL
 - spatialite.h, 97
- load_zip_dbf
 - spatialite.h, 98
- load_zip_shapefile
 - spatialite.h, 98
- math_llabs
 - spatialite.h, 100
- math_round
 - spatialite.h, 100

- remove_duplicated_rows
 - spatialite.h, 100
- remove_duplicated_rows_ex
 - spatialite.h, 101
- remove_duplicated_rows_ex2
 - spatialite.h, 101
- reset_wfs_http_connection
 - gg_wfs.h, 566
- sanitize_all_geometry_columns
 - spatialite.h, 102
- sanitize_all_geometry_columns_r
 - spatialite.h, 103
- sanitize_geometry_column
 - spatialite.h, 104
- sanitize_geometry_column_r
 - spatialite.h, 105
- spatial_ref_sys_init
 - spatialite.h, 106
- spatial_ref_sys_init2
 - spatialite.h, 106
- spatialite.h
 - check_all_geometry_columns, 60
 - check_all_geometry_columns_r, 61
 - check_duplicated_rows, 62
 - check_geometry_column, 62
 - check_geometry_column_r, 63
 - disable_tiny_point, 64
 - dump_dbf, 64
 - dump_dbf_ex, 65
 - dump_dbf_ex2, 65
 - dump_geojson, 66
 - dump_geojson2, 67
 - dump_geojson_ex, 68
 - dump_kml, 69
 - dump_kml_ex, 69
 - dump_shapefile, 70
 - dump_shapefile_ex, 71
 - dump_shapefile_ex2, 71
 - elementary_geometries, 72
 - elementary_geometries_ex, 73
 - elementary_geometries_ex2, 74
 - elementary_geometries_ex3, 74
 - enable_tiny_point, 75
 - gaia_create_routing, 76
 - gaia_create_routing_nodes, 77
 - gaiaCreateMetaCatalogTables, 77
 - gaiaCutter, 78
 - gaiaDropTable, 79
 - gaiaDropTable5, 79
 - gaiaDropTableEx, 80
 - gaiaDropTableEx2, 81
 - gaiaDropTableEx3, 81
 - gaiaFreeVectorLayersList, 82
 - gaiaGetLayerExtent, 82
 - gaiaGetVectorLayersList, 83
 - gaiaRenameColumn, 84
 - gaiaRenameTable, 85
 - gaiaStatisticsInvalidate, 86
 - gaiaUpdateMetaCatalogStatistics, 86
 - gaiaUpdateMetaCatalogStatisticsFromMaster, 87
 - insert_epsg_srid, 87
 - is_kml_constant, 88
 - is_tiny_point_enabled, 88
 - load_dbf, 89
 - load_dbf_ex, 89
 - load_dbf_ex2, 90
 - load_dbf_ex3, 91
 - load_geojson, 92
 - load_shapefile, 92
 - load_shapefile_ex, 93
 - load_shapefile_ex2, 95
 - load_shapefile_ex3, 96
 - load_XL, 97
 - load_zip_dbf, 98
 - load_zip_shapefile, 98
 - math_llabs, 100
 - math_round, 100
 - remove_duplicated_rows, 100
 - remove_duplicated_rows_ex, 101
 - remove_duplicated_rows_ex2, 101
 - sanitize_all_geometry_columns, 102
 - sanitize_all_geometry_columns_r, 103
 - sanitize_geometry_column, 104
 - sanitize_geometry_column_r, 105
 - spatial_ref_sys_init, 106
 - spatial_ref_sys_init2, 106
 - spatialite_alloc_connection, 107
 - spatialite_cleanup, 107
 - spatialite_cleanup_ex, 107
 - spatialite_finalize_topologies, 108
 - spatialite_init, 108
 - spatialite_init_ex, 109
 - spatialite_init_geos, 109
 - spatialite_initialize, 109
 - spatialite_set_silent_mode, 110
 - spatialite_set_verbode_mode, 110
 - spatialite_shutdown, 110
 - spatialite_target_cpu, 111
 - spatialite_version, 111
 - srid_get_axis, 111
 - srid_get_datum, 112
 - srid_get_prime_meridian, 112
 - srid_get_projection, 113
 - srid_get_spheroid, 113
 - srid_get_unit, 114
 - srid_has_flipped_axes, 114
 - srid_is_geographic, 115
 - srid_is_projected, 115
 - update_layer_statistics, 116
- spatialite_alloc_connection
 - spatialite.h, 107
- spatialite_cleanup
 - spatialite.h, 107
- spatialite_cleanup_ex
 - spatialite.h, 107
- spatialite_finalize_topologies

spatialite.h, 108
spatialite_init
 spatialite.h, 108
spatialite_init_ex
 spatialite.h, 109
spatialite_init_geos
 spatialite.h, 109
spatialite_initialize
 spatialite.h, 109
spatialite_set_silent_mode
 spatialite.h, 110
spatialite_set_verbode_mode
 spatialite.h, 110
spatialite_shutdown
 spatialite.h, 110
spatialite_target_cpu
 spatialite.h, 111
spatialite_version
 spatialite.h, 111
src/headers/spatialite.h, 55
src/headers/spatialite/gaiaaux.h, 116
src/headers/spatialite/gaiaexif.h, 134
src/headers/spatialite/gaiageo.h, 153
src/headers/spatialite/gg_advanced.h, 153
src/headers/spatialite/gg_const.h, 326
src/headers/spatialite/gg_core.h, 338
src/headers/spatialite/gg_dxf.h, 419
src/headers/spatialite/gg_dynamic.h, 437
src/headers/spatialite/gg_formats.h, 450
src/headers/spatialite/gg_mbr.h, 519
src/headers/spatialite/gg_structs.h, 541
src/headers/spatialite/gg_wfs.h, 550
src/headers/spatialite/gg_xml.h, 566
srid_get_axis
 spatialite.h, 111
srid_get_datum
 spatialite.h, 112
srid_get_prime_meridian
 spatialite.h, 112
srid_get_projection
 spatialite.h, 113
srid_get_spheroid
 spatialite.h, 113
srid_get_unit
 spatialite.h, 114
srid_has_flipped_axes
 spatialite.h, 114
srid_is_geographic
 spatialite.h, 115
srid_is_projected
 spatialite.h, 115

update_layer_statistics
 spatialite.h, 116

vrttxt_column_header, 51
vrttxt_line, 51
vrttxt_reader, 52
vrttxt_row, 53
vrttxt_row_block, 54