SpatiaLite

4.3.0

Generated by Doxygen 1.8.9.1

Wed Jul 1 2015 09:06:03

Contents

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 lighweight:

- a single ligthweight 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 powerfull 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 Postgre↩ SQL + PostGIS.
The main difference between them isn't in powerness, but mainly relies on architecture:

- PostgrSQL + 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 envoronments.
- 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 reccomended, 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* approch 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.
  Theorically, 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, powerfull, 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 -lpsatialite -lsqlite -lgeos_c -lgeos \
    -lproj -lfreexl -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

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Data Structure Documentation

4.1 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

4.1.1 Detailed Description

wrapper for DXF Arc object

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

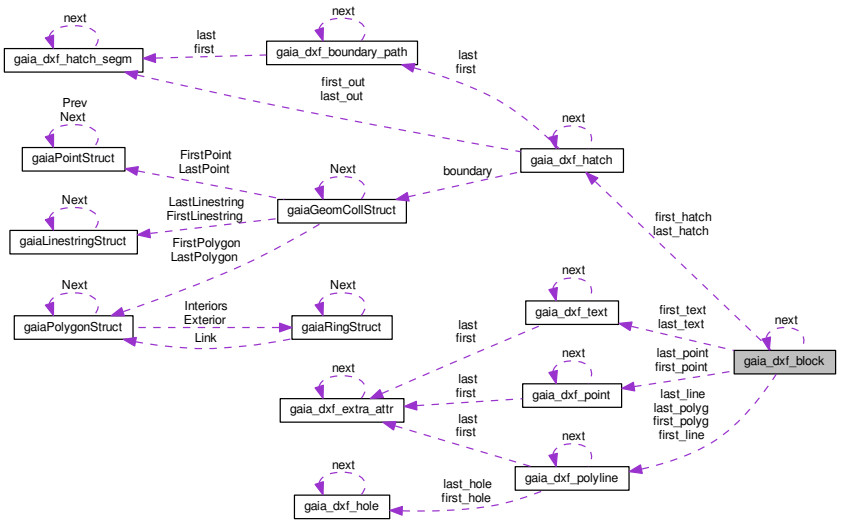
- src/headers/spatialite/gg_dxf.h

4.2 gaia_dxf_block Struct Reference

wrapper for DXF Block object

`#include <gg_dxf.h>`

Collaboration diagram for gaia_dxf_block:



Data Fields

- int hasInsert

  Boolean flag: this block is referenced by some Insert.
- char * layer_name

  pointer to Layer Name string
- char * block_id

  pointer to Block ID 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]
- 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]

4.2.1 Detailed Description

wrapper for DXF Block object

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

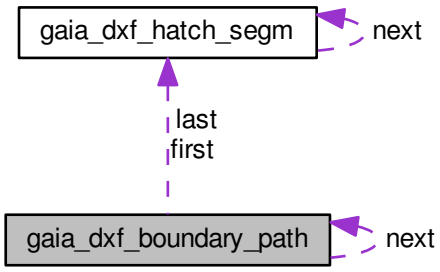
- src/headers/spatialite/gg_dxf.h

4.3 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]

4.3.1 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

4.4 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

4.4.1 Detailed Description

wrapper for DXF Circle object

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

- src/headers/spatialite/gg_dxf.h

4.5 gaia_dxf_extra_attr Struct Reference

wrapper for DXF Extra Attribute object

`#include <gg_dxf.h>`

Collaboration diagram for gaia_dxf_extra_attr:

Data Fields

- char * key
  pointer to Extra Attribute Key value
- char * value
  pointer to Extra Attribute Value string
- struct gaia_dxf_extra_attr * next
  pointer to next item [linked list]

4.5.1 Detailed Description

wrapper for DXF Extra Attribute object

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

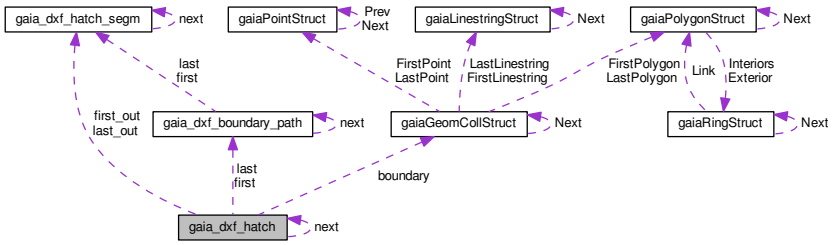
- src/headers/spatialite/gg_dxf.h

4.6 gaia_dxf_hatch Struct Reference

wrapper for DXF Pattern Hatch object

`#include <gg_dxf.h>`

Collaboration diagram for gaia_dxf_hatch:



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]

4.6.1 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

4.7 gaia_dxf_hatch_segm Struct Reference

wrapper for DXF Pattern Segment object

`#include <gg_dxf.h>`

Collaboration diagram for gaia_dxf_hatch_segm:

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]

4.7.1 Detailed Description

wrapper for DXF Pattern Segment object

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

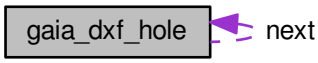
- src/headers/spatialite/gg_dxf.h

4.8 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]

4.8.1 Detailed Description

wrapper for DXF Polygon interior hole object

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

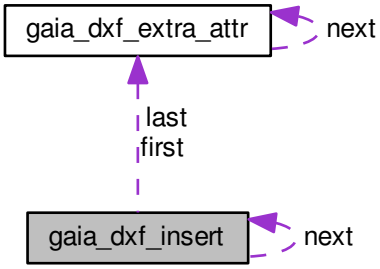
- src/headers/spatialite/gg_dxf.h

4.9 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]

4.9.1 Detailed Description

wrapper for DXF Insert object

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

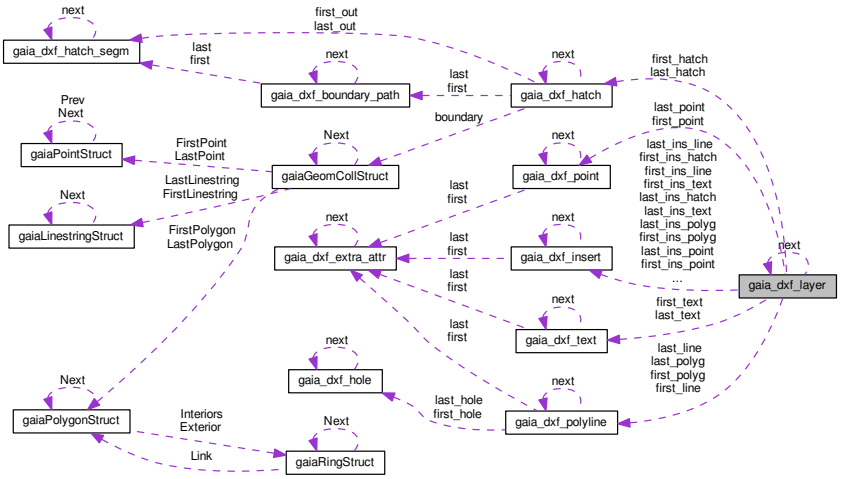
- src/headers/spatialite/gg_dxf.h

4.10 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 lasst 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]
- 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]

4.10.1 Detailed Description

wrapper for DXF Layer object

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

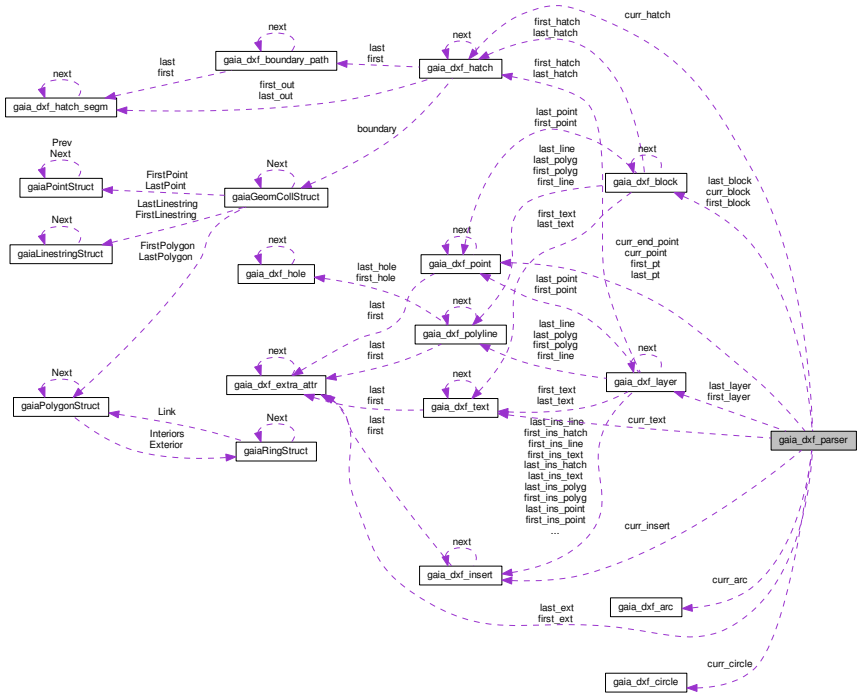
- src/headers/spatialite/gg_dxf.h

4.11 gaia_dxf_parser Struct Reference

wrapper for DXF Parser object

`#include <gg_dxf.h>`

Collaboration diagram for gaia_dxf_parser:



Data Fields

- char * filename

  OUT: origin/input filename.
- gaiaDxfLayerPtr first_layer

  OUT: pointer to first DXF Layer object [linked list].
- gaiaDxfLayerPtr last_layer

  OUT: pointer to last DXF Layer object [linked list].
- gaiaDxfBlockPtr first_block

  OUT: pointer to first DXF Block object [linked list].
- gaiaDxfBlockPtr last_block

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

  IN: parser option - dimension handlig.
- 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

4.11.1 Detailed Description

wrapper for DXF Parser object

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

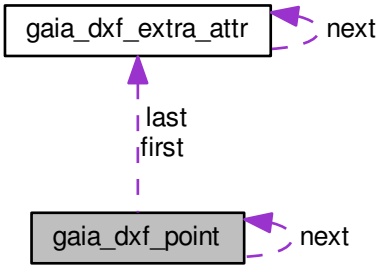
- src/headers/spatialite/gg_dxf.h

4.12 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]

4.12.1 Detailed Description

wrapper for DXF Point object

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

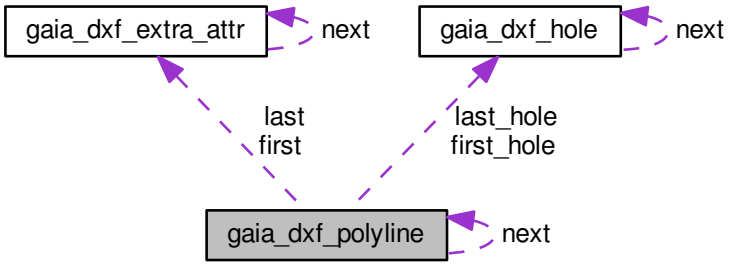
- src/headers/spatialite/gg_dxf.h

4.13 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]

4.13.1 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:

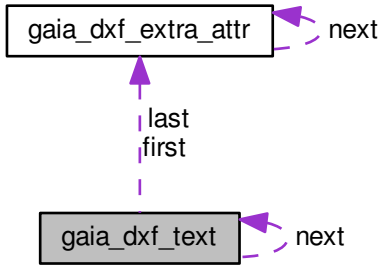
- src/headers/spatialite/gg_dxf.h

4.14 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]

4.14.1 Detailed Description

wrapper for DXF Text object

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

- src/headers/spatialite/gg_dxf.h

4.15 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.

4.15.1 Detailed Description

wrapper for DXF Write object

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

- src/headers/spatialite/gg_dxf.h

4.16 gaiaAttributeFieldDoubleRangeInfos Struct Reference

Attribute/Field Double range infos.

`#include <gg_structs.h>`

Data Fields

- double MinValue
  Minimum value.
- double MaxValue
  Maximum value.

4.16.1 Detailed Description

Attribute/Field Double range infos.

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

- src/headers/spatialite/gg_structs.h

4.17 gaiaAttributeFieldIntRangeInfos Struct Reference

Attribute/Field Integer range infos.

`#include <gg_structs.h>`

Data Fields

- sqlite3_int64 MinValue

  Minimum value.

- sqlite3_int64 MaxValue

  Maximum value.

4.17.1 Detailed Description

Attribute/Field Integer range infos.

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

- src/headers/spatialite/gg_structs.h

4.18 gaiaAttributeFieldMaxSizeInfos Struct Reference

Attribute/Field MaxSize/Length infos.

`#include <gg_structs.h>`

Data Fields

- int MaxSize

  MaxSize / MaxLength.

4.18.1 Detailed Description

Attribute/Field MaxSize/Length infos.

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

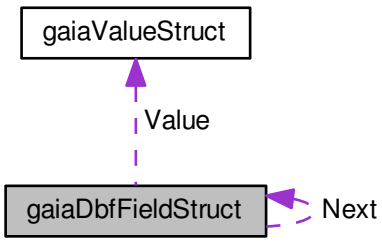
- src/headers/spatialite/gg_structs.h

4.19 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]

4.19.1 Detailed Description

Container for DBF field.

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

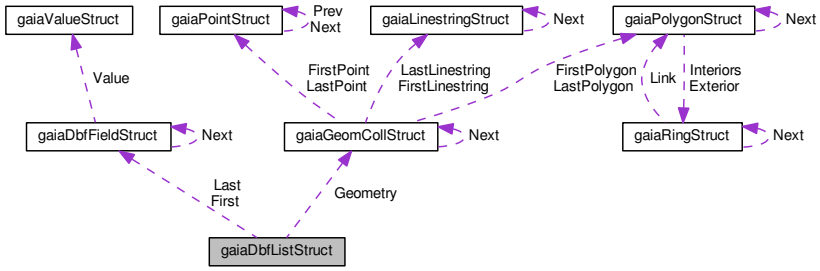
- src/headers/spatialite/gg_structs.h

4.20 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]

4.20.1 Detailed Description

Container for a list of DBF fields.

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

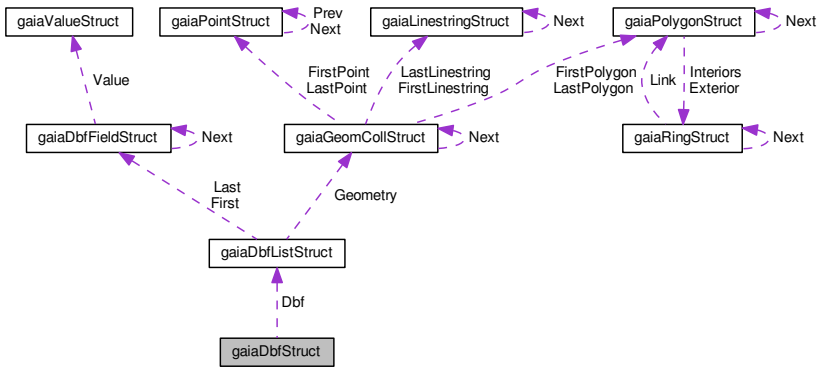
- src/headers/spatialite/gg_structs.h

4.21 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 * flDbf

  FILE handle.
- 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)

4.21.1 Detailed Description

Container for DBF file handling.

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

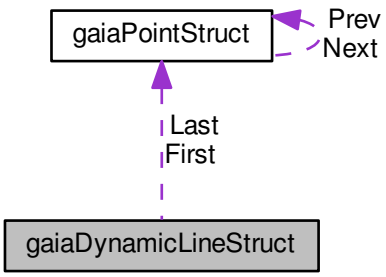
- src/headers/spatialite/gg_structs.h

4.22 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]

4.22.1 Detailed Description

Container for dynamically growing line/ring.

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

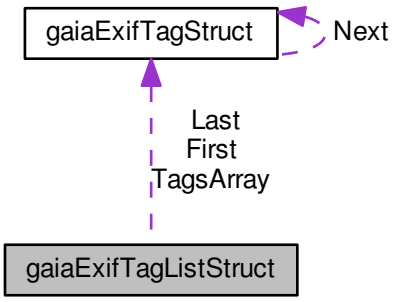
- src/headers/spatialite/gg_structs.h

4.23 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

4.23.1 Detailed Description

Container for a list of EXIF tags.

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

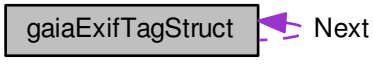
- src/headers/spatialite/gaiaexif.h

4.24 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

4.24.1 Detailed Description

Container for an EXIF tag.

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

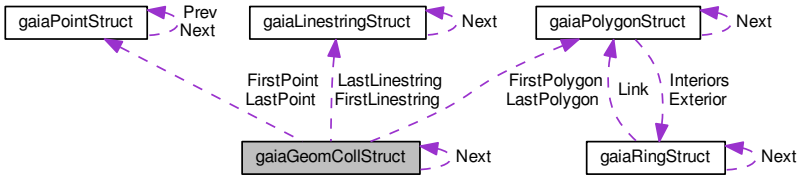
- src/headers/spatialite/gaiaexif.h

4.25 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]

4.25.1 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:

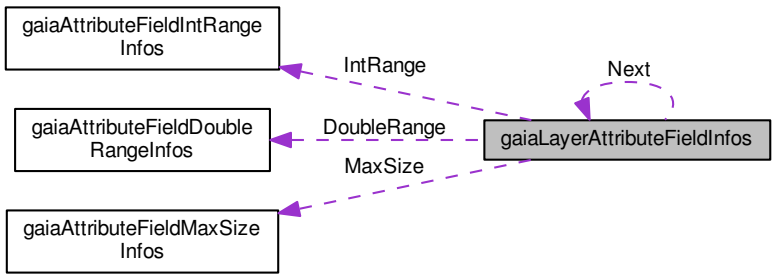
- src/headers/spatialite/gg_structs.h

4.26 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)

4.26.1 Detailed Description

LayerAttributeField infos.

Examples:

demo5.c.

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

- src/headers/spatialite/gg_structs.h

4.27 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.

4.27.1 Detailed Description

Layer Auth infos.

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

- src/headers/spatialite/gg_structs.h

4.28 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.

4.28.1 Detailed Description

Layer Extent infos.

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

- src/headers/spatialite/gg_structs.h

4.29 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 X.

- int DimensionModel

  one of GAIA_XY, GAIA_XY_Z, GAIA_XY_M, GAIA_XY_ZM
- struct gaiaLinestringStruct * Next

  pointer to next item [linked list]

4.29.1 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

4.30 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

4.30.1 Detailed Description

Container for dynamically growing output buffer.

Examples:

demo2.c.

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

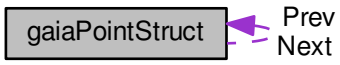
- src/headers/spatialite/gg_structs.h

4.31 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]

4.31.1 Detailed Description

Container for OGC POINT Geometry.

Examples:

demo2.c.

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

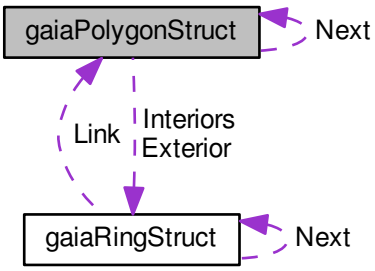
- src/headers/spatialite/gg_structs.h

4.32 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]

4.32.1 Detailed Description

Container for OGC POLYGON Geometry.

Examples:

demo2.c.

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

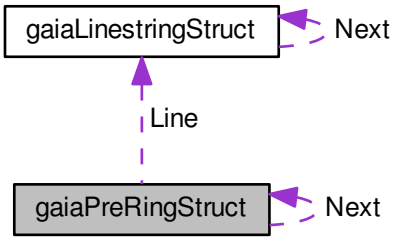
- src/headers/spatialite/gg_structs.h

4.33 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]

4.33.1 Detailed Description

Container similar to LINESTRING [internally used].

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

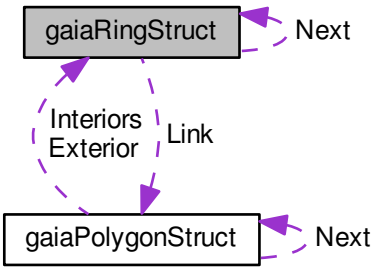
- src/headers/spatialite/gg_structs.h

4.34 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

4.34.1 Detailed Description

Container for OGC RING Geometry.

Examples:

demo2.c.

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

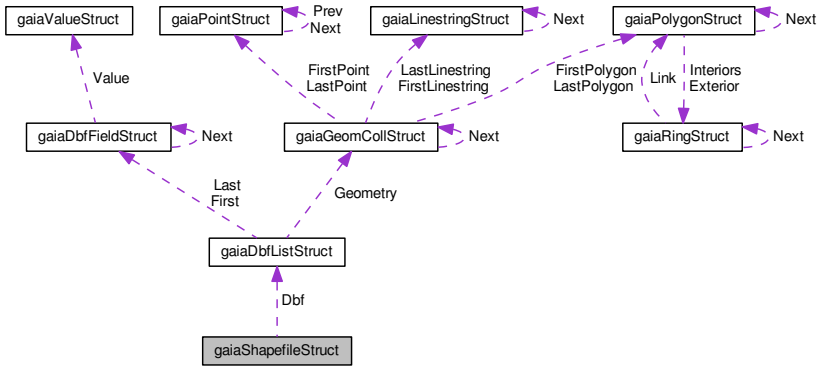
- src/headers/spatialite/gg_structs.h

4.35 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.
- 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.

4.35.1 Detailed Description

Container for SHP file handling.

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

- src/headers/spatialite/gg_structs.h

4.36 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.

4.36.1 Detailed Description

Container for variant (multi-type) value.

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

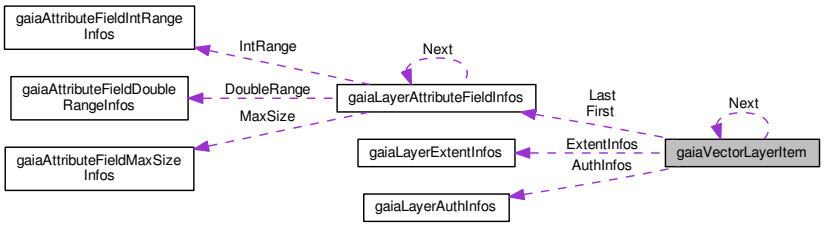
- src/headers/spatialite/gg_structs.h

4.37 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_P↩ OLYGON, GAIA_VECTOR_MULTIPOINT, GAIA_VECTOR_MULTILINESTRING, GAIA_VECTOR_MULTIPOLYG↩ ON, 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_S↩ PATIAL_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)

4.37.1 Detailed Description

Vector Layer item.

Examples:

demo5.c.

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

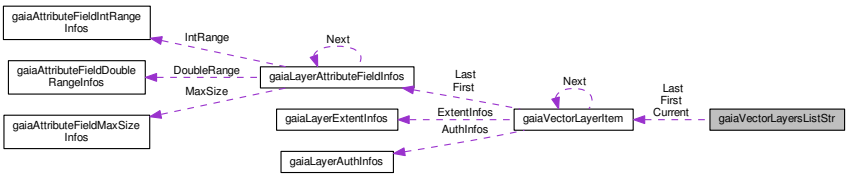
- src/headers/spatialite/gg_structs.h

4.38 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

4.38.1 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

4.39 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

4.39.1 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

4.40 vrttxt_line Struct Reference

Container for Virtual Text record (line)

`#include <gg_structs.h>`

Data Fields

- 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

4.40.1 Detailed Description

Container for Virtual Text record (line)

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

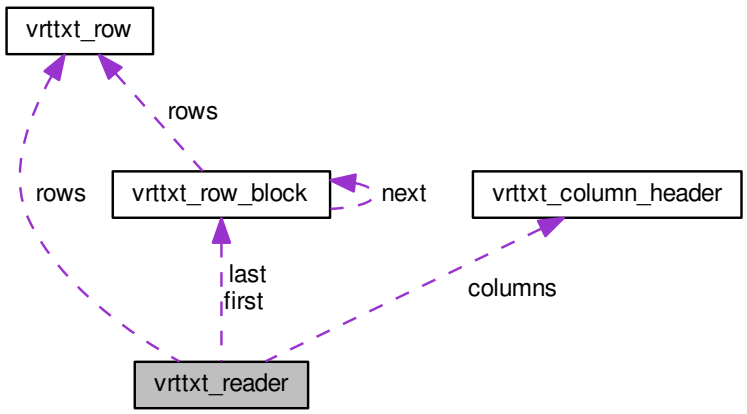
- src/headers/spatialite/gg_structs.h

4.41 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

4.41.1 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

4.42 vrttxt_row Struct Reference

Container for Virtual Text record (line) offsets.

`#include <gg_structs.h>`

Data Fields

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

4.42.1 Detailed Description

Container for Virtual Text record (line) offsets.

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

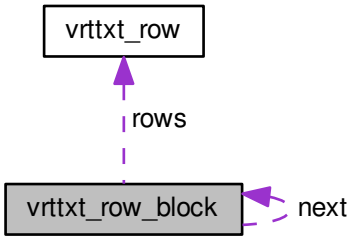
- src/headers/spatialite/gg_structs.h

4.43 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]

4.43.1 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 5

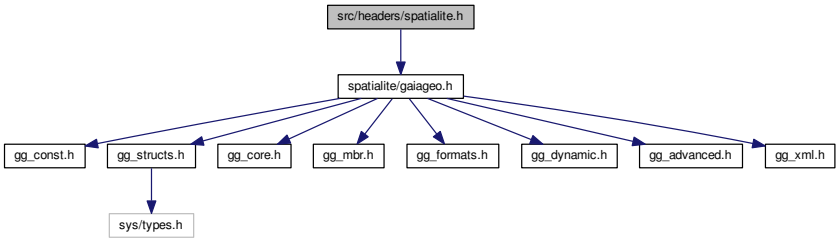
File Documentation

5.1 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_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 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 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_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 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 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 *multiId)

  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 *multiId, 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 *multiId, 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.
- 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.
- 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_metacalog_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 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 **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)

5.1.1 Detailed Description

Main SpatiaLite header file.

5.1.2 Function Documentation

5.1.2.1 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

sqlite	handle to current DB connection
output_dir	pathname of the directory to be created for report-files
n_invalids	if this variable is not NULL on successful completion will contain the total number of invalid Geometries found
err_msg	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

5.1.2.2 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

p_cache	a memory pointer returned by spatialite_alloc_connection()
sqlite	handle to current DB connection
output_dir	pathname of the directory to be created for report-files
n_invalids	if this variable is not NULL on successful completion will contain the total number of invalid Geometries found

err_msg	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

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

Checks for duplicated rows into the same table.

Parameters

sqlite	handle to current DB connection
table	name of the table to be checked
dupl_count	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 exacly the same

5.1.2.4 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

sqlite	handle to current DB connection
table	name of the table
geometry	name of the column to be checked
report_path	pathname of the report-file
n_rows	if this variable is not NULL on successful completion will contain the total number of rows found into the checkeck table

n_invalids	if this variable is not NULL on successful completion will contain the total number of invalid Geometries found into the checkeck table
err_msg	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

5.1.2.5 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

p_cache	a memory pointer returned by spatialite_alloc_connection()
sqlite	handle to current DB connection
table	name of the table
geometry	name of the column to be checked
report_path	pathname of the report-file
n_rows	if this variable is not NULL on successful completion will contain the total number of rows found into the checkeck table
n_invalids	if this variable is not NULL on successful completion will contain the total number of invalid Geometries found into the checkeck table
err_msg	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

5.1.2.6 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

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

See also

dump_dbf_ex

Returns

0 on failure, any other value on success

5.1.2.7 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

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

See also

dump_dbf

Returns

0 on failure, any other value on success

5.1.2.8 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.

Parameters

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

See also

dump_geojson_rx

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

5.1.2.9 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.

Parameters

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

See also

dump_geojson

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

5.1.2.10 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

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

See also

dump_kml_ex

Returns

0 on failure, any other value on success

5.1.2.11 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

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

See also

dump_kml

Returns

0 on failure, any other value on success

5.1.2.12 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

sqlite	handle to current DB connection
table	the name of the table to be exported
column	the name of the geometry column

shp_path	pathname of the Shapefile to be exported (no suffix)
charset	a valid GNU ICONV charset to be used for DBF text strings
geom_type	"POINT", "LINESTRING", "POLYGON", "MULTIPOLYGON" or NULL
verbose	if TRUE a short report is shown on stderr
rows	on completion will contain the total number of exported rows
err_msg	on completion will contain an error message (if any)

Returns

0 on failure, any other value on success

5.1.2.13 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

sqlite	handle to current DB connection
inTable	name of the input table
geometry	name of the Geometry column
outTable	name of the output table to be created
pKey	name of the Primary Key column in the output table
multiId	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, MULTIPOL↩ YGON or GEOMETRYCOLLECTION), then many rows are inserted into the output table: each single row will contain the same attributes and an elementaty Geometry. All the rows created by expanding the same input row will expose the same value in the "multiId" column.

5.1.2.14 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

sqlite	handle to current DB connection
inTable	name of the input table
geometry	name of the Geometry column
outTable	name of the output table to be created
pKey	name of the Primary Key column in the output table
multiId	name of the column identifying origins in the output table
rows	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, MULTIPOL↩ YGON or GEOMETRYCOLLECTION), then many rows are inserted into the output table: each single row will contain the same attributes and an elementaty Geometry. All the rows created by expanding the same input row will expose the same value in the "multiId" column.

5.1.2.15 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

sqlite	handle to current DB connection
inTable	name of the input table
geometry	name of the Geometry column
outTable	name of the output table to be created
pKey	name of the Primary Key column in the output table
multiId	name of the column identifying origins in the output table
rows	on completion will contain the total number of inserted rows
transaction	boolena; 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, MULTIPOL↩ YGON or GEOMETRYCOLLECTION), then many rows are inserted into the output table: each single row will contain the same attributes and an elementaty Geometry. All the rows created by expanding the same input row will expose the same value in the "multiId" column.

5.1.2.16 SPATIALITE_DECLARE int gaiaCreateMetaCatalogTables (sqlite3 * *handle*)

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

Parameters

handle	SQLite handle to current DB connection.

Returns

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

See also

gaiaUpdateMetaCatalogStatistics, gaiaUpdateMetaCatalogStatisticsFromMaster

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

Drops a layer-table, removing any related dependency.

Parameters

sqlite	handle to current DB connection
table	name of the table to be removed

Note

this function will drop a SpatialTable, SpatialView or VirtualShape being properly registered within the Metadata 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

Note

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

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

Drops a layer-table, removing any related dependency.

Parameters

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

Note

this function will drop a SpatialTable, SpatialView or VirtualShape being properly registered within the Metadata 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

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

Drops a layer-table, removing any related dependency.

Parameters

sqlite	handle to current DB connection
prefix	schema prefix identifying the target DB "main" always identifies the main DB (primary, not Attached).
table	name of the table to be removed
transaction	boolena; 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 Metadata 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

5.1.2.20 SPATIALITE_DECLARE void gaiaFreeVectorLayersList (gaiaVectorLayersListPtr *ptr*)

Destroys a VectorLayersList object.

Parameters

ptr	pointer to the VectorLayersList object to be destroyed

See also

gaiaGetVectorLayersList

Examples:

demo5.c.

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

Queries the Metadata tables returning the Layer Full Extent.

Parameters

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

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

Queries the Metadata tables supporting Vector Layers.

Parameters

handle	SQLite handle to current DB connection.
table	VectorLayer Table (or View, or VirtualShape).
geometry	Geometry Column name.
mode	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; 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.

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

Immediately and unconditionally invalidates the already existing Statistics.

Parameters

handle	SQLite handle to current DB connection.
table	VectorLayer Table (or View, or VirtualShape).
geometry	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.

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

Updates the "splite_metacatalog_statistics" table.

Parameters

handle	SQLite handle to current DB connection.
table	name of the table to be processed.
column	name of the column to be processed.

Returns

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

See also

gaiaCreateMetaCatalogTables, gaiaUpdateMetaCatalogStatisticsFromMaster

5.1.2.25 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

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

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

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

Parameters

sqlite	handle to current DB connection
srid	the SRID value uniquely identifying the required EPSG definition

Returns

0 on failure, any other value on success

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

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

Parameters

sqlite	handle to current DB connection
table	the table to be checked
column	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

5.1.2.28 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

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

See also

load_dbf_ex, load_dbf_ex2

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

5.1.2.29 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

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

See also

load_dbf, load_dbf_ex2

Returns

0 on failure, any other value on success

5.1.2.30 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

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

See also

load_dbf, load_dbf_ex

Returns

0 on failure, any other value on success

5.1.2.31 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

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

Note

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

5.1.2.32 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

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

spatial_index	if TRUE an R∗Tree Spatial Index will be created
rows	on completion will contain the total number of imported rows
err_msg	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

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.

5.1.2.33 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

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

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.

5.1.2.34 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

sqlite	handle to current DB connection
path	pathname of the spreadsheet file to be imported
table	the name of the table to be created
worksheetIndex	the index identifying the worksheet to be imported
first_titles	if TRUE the first line is assumed to contain column names
rows	on completion will contain the total number of actually exported rows
err_msg	on completion will contain an error message (if any)

Returns

0 on failure, any other value on success

5.1.2.35 SPATIALITE_DECLARE sqlite3_int64 math_llabs (sqlite3_int64 *value*)

A portable replacement for C99 llabs()

Parameters

value	a 64 bit integer value

Returns

the corresponding absolute value

5.1.2.36 SPATIALITE_DECLARE double math_round (double *value*)

A portable replacement for C99 round()

Parameters

value	a double value

Returns

the nearest integeral value

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

Remove duplicated rows from a table.

Parameters

sqlite	handle to current DB connection
table	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 occurence will be preserved, then deleting any further occurrence.

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

Remove duplicated rows from a table.

Parameters

sqlite	handle to current DB connection
table	name of the table to be cleaned
removed	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 occurence will be preserved, then deleting any further occurrence.

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

Remove duplicated rows from a table.

Parameters

sqlite	handle to current DB connection

table	name of the table to be cleaned
removed	on successful completion will contain the total count of removed duplicate rows
transaction	boolena; 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 occurence will be preserved, then deleting any further occurrence.

5.1.2.40 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

sqlite	handle to current DB connection
tmp_prefix	name-prefix for temporary tables
output_dir	pathname of the directory to be created for report-files
not_repaired	if this variable is not NULL on successful completion will contain the total count of repair failures (i.e. Geometries beyond possible repair)
err_msg	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 temprary 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

5.1.2.41 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

p_cache	a memory pointer returned by spatialite_alloc_connection()
sqlite	handle to current DB connection
tmp_prefix	name-prefix for temporary tables
output_dir	pathname of the directory to be created for report-files
not_repaired	if this variable is not NULL on successful completion will contain the total count of repair failures (i.e. Geometries beyond possible repair)
err_msg	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 temprary 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

5.1.2.42 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

sqlite	handle to current DB connection
table	name of the table
geometry	name of the column to be checked
tmp_table	name of the temporary table
report_path	pathname of the report-file
n_invalids	if this variable is not NULL on successful completion will contain the total number of invalid Geometries found into the sanitize table
n_repaired	if this variable is not NULL on successful completion will contain the total number of repaired Geometries
n_discarded	if this variable is not NULL on successful completion will contain the total number of repaired Geometries (by discarding fragments)
n_failures	if this variable is not NULL on successful completion will contain the total number of repair failures (i.e. Geometries beyond possible repair)
err_msg	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 temprary 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

5.1.2.43 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

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

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

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

Parameters

sqlite	handle to current DB connection
verbose	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 functions is now deprecated, and will simply call spatial_ref_sys_init2(sqlite, GAIA_EPSG_↩ ANY, verbose).

5.1.2.45 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

sqlite	handle to current DB connection
mode	can be one of GAIA_EPSG_ANY, GAIA_EPSG_NONE or GAIA_EPSG_WGS84_ONLY
verbose	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.

5.1.2.46 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.

5.1.2.47 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

5.1.2.48 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

ptr	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.

5.1.2.49 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

verbose	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.

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

Initializes a SpatiaLite connection.

Parameters

db_handle	handle to the current SQLite connection
ptr	a memory pointer returned by spatialite_alloc_connection()
verbose	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.

5.1.2.51 SPATIALITE_DECLARE void spatialite_init_geos (void)

Initializes the GEOS library.

Note

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

5.1.2.52 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.

5.1.2.53 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.

5.1.2.54 SPATIALITE_DECLARE const char* spatialite_target_cpu (void)

Return the target CPU name.

Returns

the target CPU string.

5.1.2.55 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.

5.1.2.56 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

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

Returns

the reqested name on succes, NULL on failure

Note

you are responsible for freeing the returned name.

5.1.2.57 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

sqlite	handle to current DB connection
srid	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.

5.1.2.58 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

sqlite	handle to current DB connection
srid	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.

5.1.2.59 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

sqlite	handle to current DB connection
srid	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.

5.1.2.60 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

sqlite	handle to current DB connection
srid	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.

5.1.2.61 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

sqlite	handle to current DB connection
srid	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.

5.1.2.62 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

sqlite	handle to current DB connection
srid	the SRID value uniquely identifying the required EPSG definition
flipped	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

5.1.2.63 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

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

Returns

0 on failure, any other value on success

5.1.2.64 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

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

Returns

0 on failure, any other value on success

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

Updates the LAYER_STATICS metadata table.

Parameters

sqlite	handle to current DB connection
table	name of the table to be processed
column	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

5.2 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 gaiaIsReservedSqliteName (const char *name)

  Checks if a name is a reserved SQLite name.
- GAIAAUX_DECLARE int gaiaIsReservedSqlName (const char *name)

  Checks if a name is a reserved SQL name.
- GAIAAUX_DECLARE int gaiaIllegalSqlName (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 * gaiaConvertToDMS (double longitude, double latitude)

  Return a DMS string.
- GAIAAUX_DECLARE char * gaiaEncodeURL (const char *url)

  Return a percent-encoded URL.
- GAIAAUX_DECLARE char * gaiaDecodeURL (const char *encoded)

  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.

5.2.1 Detailed Description

Auxiliary/helper functions.

5.2.2 Function Documentation

5.2.2.1 GAIAAUX_DECLARE void gaiaCleanSqlString (char * *value*)

deprecated function

Parameters

value	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.

5.2.2.2 GAIAAUX_DECLARE int gaiaConvertCharset (char ** *buf,* const char * *fromCs,* const char * *toCs*)

Converts a text string from one charset to another.

Parameters

buf	the text string to be converted
fromCs	the GNU ICONV name identifying the input charset
toCs	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.

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

Return a DMS string.

Parameters

longitude	the angle of longitude expressed in Decimal Degrees.
latitude	the angle of latitude expressed in Decimal Degrees.

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.

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

Converts a text string to UTF8.

Parameters

cvtCS	the handle identifying the UTF8 convert object (returned by a previous call to gaiaCreateU↩ TF8Converter).
buf	the input text string
len	length (in bytes) of input string
err	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.

5.2.2.5 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.

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

Creates a persistent UTF8 converter object.

Parameters

fromCS	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.

5.2.2.7 GAIAAUX_DECLARE char* gaiaDecodeURL (const char * *encoded*)

Return a clean URL from its percent-encoded representation.

Parameters

encoded	the percent-encoded 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.

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

Properly formats an SQL generic string (dequoting)

Parameters

value	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.

5.2.2.9 GAIAAUX_DECLARE char* gaiaDirNameFromPath (const char * *path*)

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

Parameters

path	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.

5.2.2.10 GAIAAUX_DECLARE char* gaiaDoubleQuotedSql (const char * *value*)

Properly formats an SQL name.

Parameters

value	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_DOUB↩ LE_QUOTE);

Remarks

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

5.2.2.11 GAIAAUX_DECLARE char* gaiaEncodeURL (const char * *url*)

Return a percent-encoded URL.

Parameters

url	the URL to be percent-encoded

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.

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

Return the FileExtension from a Path.

Parameters

path	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.

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

Return the FileName from a Path.

Parameters

path	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.

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

Return an MD5 checksum value.

Parameters

md5	the handle of the MD5 checksum object (returned by a previous call to gaiaCreateMD5↩ Checksum).

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.

5.2.2.15 GAIAAUX_DECLARE void gaiaFreeMD5Checksum (void * *md5*)

Destroys an MD5 checksum object.

Parameters

md5	the handle of the MD5 checksum object (returned by a previous call to gaiaCreateMD5↩ Checksum).

See also

gaiaCreateMD5Checksum

5.2.2.16 GAIAAUX_DECLARE void gaiaFreeUTF8Converter (void * *cvtCS*)

Destroys an UTF8 converter object.

Parameters

cvtCS	the handle identifying the UTF8 convert object (returned by a previous call to gaiaCreateU↩ TF8Converter).

See also

gaiaCreateUTF8Converter

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

Return the FullFileName from a Path.

Parameters

path	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.

5.2.2.18 GAIAAUX_DECLARE const char* gaiaGetLocaleCharset (void)

Retrieves the Locale Charset.

Returns

the GNU ICONV name identifying the locale charset

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

Checks if a name is an illegal SQL name.

Parameters

name	the name to be checked

Returns

0 if no: any other value if yes

See also

gaiaIsReservedSqliteName, gaiaIsReservedSqlName

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

SQL log: statement start.

Parameters

sqlite	handle of the current DB connection
user_agent	name of the invoking application, e.g. "spatialite_gui" or "spatialite CLI"
utf8Sql	the SQL statement bein executed
sqllog_pk	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.

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

Checks if a name is a reserved SQLite name.

Parameters

name	the name to be checked

Returns

0 if no: any other value if yes

See also

gaiaIsReservedSqlName, gaiaIllegalSqlName

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

Checks if a name is a reserved SQL name.

Parameters

name	the name to be checked

Returns

0 if no: any other value if yes

See also

gaiaIsReservedSqliteName, gaiaIllegalSqlName

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

Return longitude and latitude angles from a DMS string.

Parameters

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

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

Properly formats an SQL generic string.

Parameters

value	the string to be formatted
quote	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.

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

Properly formats an SQL text constant.

Parameters

value	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_SINGL↩ E_QUOTE);

Remarks

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

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

Updates an MD5 checksum object.

Parameters

md5	the handle of the MD5 checksum object (returned by a previous call to gaiaCreateMD5↩ Checksum).
blob	an arbitrary sequence of binary data
blob_size	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 M↩ D5 object. In this case the final MD5 checksum returned by gaiaGetMD5Checsum will be the total checksum for any data processed by the MD5 object since its initialization.

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

SQL log: statement start.

Parameters

sqlite	handle of the current DB connection
sqllog_pk	the Primary Key identifying the corresponding Log event. expected to be exactely the same returned by the most recent call to gaiaInsertIntoSqlLog()
success	expected to be TRUE if the SQL statement was successfully executed.
errMsg	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.

5.3 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 containt 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
- #define GAIA_TIFF_BLOB 9

  this BLOB does actually contain a TIFF image
- #define GAIA_WEBP_BLOB 10

  this BLOB does actually contain a WebP image
- #define GAIA_JP2_BLOB 11

  this BLOB does actually contain a JP2 (Jpeg2000) image
- #define GAIA_XML_BLOB 12

  this BLOB does actually contain a SpatiaLite XmlBLOB
- #define GAIA_GPB_BLOB 13

  this BLOB does actually contain a GPKG Geometry
- #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.

5.3.1 Detailed Description

EXIF/image: supporting functions and constants.

5.3.2 Typedef Documentation

5.3.2.1 typedef gaiaExifTagList* gaiaExifTagListPtr

Typedef for EXIF tag structure.

See also

gaiaExifTagListStruct

5.3.2.2 typedef gaiaExifTag* gaiaExifTagPtr

Typedef for EXIF tag structure.

See also

gaiaExifTagStruct

5.3.3 Function Documentation

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

Return a BYTE value from an EXIF tag.

Parameters

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

Returns

the BYTE value

See also

gaiaGetExifTagById, gaiaGetExifGpsTagById, gaiaGetExifTagByName, gaiaExifTagGetValueType, gaiaExifTagGetNumValues

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

Return a DOUBLE value from an EXIF tag.

Parameters

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

Returns

the DOUBLE value

See also

gaiaGetExifTagById, gaiaGetExifGpsTagById, gaiaGetExifTagByName, gaiaExifTagGetValueType, gaiaExifTagGetNumValues

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

Return a FLOAT value from an EXIF tag.

Parameters

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

Returns

the FLOAT value

See also

gaiaGetExifTagById, gaiaGetExifGpsTagById, gaiaGetExifTagByName, gaiaExifTagGetValueType, gaiaExif↩TagGetNumValues

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

Return a human readable description from an EXIF tag.

Parameters

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

See also

gaiaGetExifTagById, gaiaGetExifGpsTagById, gaiaGetExifTagByName

5.3.3.5 GAIAEXIF_DECLARE unsigned short gaiaExifTagGetId (const gaiaExifTagPtr *tag*)

Return the Tag ID from an EXIF tag.

Parameters

tag	pointer to an EXIF tag

Returns

the Tag ID

See also

gaiaGetExifTagById, gaiaGetExifGpsTagById, gaiaGetExifTagByName

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

Return a LONG value from an EXIF tag.

Parameters

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

Returns

the LONG value

See also

gaiaGetExifTagById, gaiaGetExifGpsTagById, gaiaGetExifTagByName, gaiaExifTagGetValueType, gaiaExif↩ TagGetNumValues

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

Return the Tag Name from an EXIF tag.

Parameters

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

See also

gaiaGetExifTagById, gaiaGetExifGpsTagById, gaiaGetExifTagByName

5.3.3.8 GAIAEXIF_DECLARE unsigned short gaiaExifTagGetNumValues (const gaiaExifTagPtr *tag*)

Return the total count of values from an EXIF tag.

Parameters

tag	pointer to an EXIF tag

Returns

the number of available values

See also

gaiaGetExifTagById, gaiaGetExifGpsTagById, gaiaGetExifTagByName

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

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

Parameters

tag	pointer to an EXIF tag.
ind	value index [first value has index 0].

ok	on completion will contain 0 on failure: any other value on success.

Returns

the RATIONAL [numerator] value

See also

gaiaGetExifTagById, gaiaGetExifGpsTagById, gaiaGetExifTagByName, gaiaExifTagGetValueType, gaiaExif↩TagGetNumValues

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

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

Parameters

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

Returns

the RATIONAL [denominator] value

See also

gaiaGetExifTagById, gaiaGetExifGpsTagById, gaiaGetExifTagByName, gaiaExifTagGetValueType, gaiaExif↩TagGetNumValues

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

Return a RATIONAL value from an EXIF tag.

Parameters

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

Returns

the RATIONAL value

See also

gaiaGetExifTagById, gaiaGetExifGpsTagById, gaiaGetExifTagByName, gaiaExifTagGetValueType, gaiaExif↩TagGetNumValues

5.3.3.12 GAIAEXIF_DECLARE unsigned short gaiaExifTagGetShortValue (const gaiaExifTagPtr *tag*, const int *ind*, int * *ok*)

Return a SHORT value from an EXIF tag.

Parameters

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

Returns

the SHORT value

See also

gaiaGetExifTagById, gaiaGetExifGpsTagById, gaiaGetExifTagByName, gaiaExifTagGetValueType, gaiaExifTagGetNumValues

5.3.3.13 GAIAEXIF_DECLARE int gaiaExifTagGetSignedLongValue (const gaiaExifTagPtr *tag*, const int *ind*, int * *ok*)

Return a Signed LONG value from an EXIF tag.

Parameters

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

Returns

the Signed LONG value

See also

gaiaGetExifTagById, gaiaGetExifGpsTagById, gaiaGetExifTagByName, gaiaExifTagGetValueType, gaiaExifTagGetNumValues

5.3.3.14 GAIAEXIF_DECLARE int gaiaExifTagGetSignedRational1Value (const gaiaExifTagPtr *tag*, const int *ind*, int * *ok*)

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

Parameters

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

Returns

the SRATIONAL [numerator] value

See also

gaiaGetExifTagById, gaiaGetExifGpsTagById, gaiaGetExifTagByName, gaiaExifTagGetValueType, gaiaExifTagGetNumValues

5.3.3.15 GAIAEXIF_DECLARE int gaiaExifTagGetSignedRational2Value (const gaiaExifTagPtr *tag*, const int *ind*, int * *ok*)

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

Parameters

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

Returns

the SRATIONAL [denominator] value

See also

gaiaGetExifTagById, gaiaGetExifGpsTagById, gaiaGetExifTagByName, gaiaExifTagGetValueType, gaiaExifTagGetNumValues

5.3.3.16 GAIAEXIF_DECLARE double gaiaExifTagGetSignedRationalValue (const gaiaExifTagPtr *tag*, const int *ind*, int * *ok*)

Return a Signed RATIONAL value from an EXIF tag.

Parameters

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

Returns

the Signed RATIONAL value

See also

gaiaGetExifTagById, gaiaGetExifGpsTagById, gaiaGetExifTagByName, gaiaExifTagGetValueType, gaiaExifTagGetNumValues

5.3.3.17 GAIAEXIF_DECLARE short gaiaExifTagGetSignedShortValue (const gaiaExifTagPtr *tag*, const int *ind*, int * *ok*)

Return a Signed SHORT value from an EXIF tag.

Parameters

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

Returns

the Signed SHORT value

See also

gaiaGetExifTagById, gaiaGetExifGpsTagById, gaiaGetExifTagByName, gaiaExifTagGetValueType, gaiaExifTagGetNumValues

5.3.3.18 GAIAEXIF_DECLARE void gaiaExifTagGetStringValue (const gaiaExifTagPtr *tag*, char * *str*, int *len*, int * *ok*)

Return a STRING value from an EXIF tag.

Parameters

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

See also

gaiaGetExifTagById, gaiaGetExifGpsTagById, gaiaGetExifTagByName, gaiaExifTagGetValueType, gaiaExif↩TagGetNumValues

5.3.3.19 GAIAEXIF_DECLARE unsigned short gaiaExifTagGetValueType (const gaiaExifTagPtr *tag*)

Return the value type for an EXIF tag.

Parameters

tag	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

5.3.3.20 GAIAEXIF_DECLARE void gaiaExifTagsFree (gaiaExifTagListPtr *tag_list*)

Destroy a list of EXIF tags.

Parameters

tag_list	the list to be destroied

See also

gaiaGetExifTags

Note

the pointer passed to this function must be one returned by a previous call to gaiaGetExifTags

5.3.3.21 GAIAEXIF_DECLARE gaiaExifTagPtr gaiaGetExifGpsTagById (const gaiaExifTagListPtr *tag_list,* const unsigned short *tag_id*)

Retrieves an EXIF-GPS tag by its Tag ID.

Parameters

tag_list	pointer to an EXIF tag list.
tag_id	the GPS Tag ID to be found

Returns

a pointer to the corresponding EXIF tag: NULL if not found

See also

gaiaGetExifTags, gaiaExifTagsFree

5.3.3.22 GAIAEXIF_DECLARE gaiaExifTagPtr gaiaGetExifTagById (const gaiaExifTagListPtr *tag_list*, const unsigned short *tag_id*)

Retrieves an EXIF tag by its Tag ID.

Parameters

tag_list	pointer to an EXIF tag list.
tag_id	the Tag ID to be found

Returns

a pointer to the corresponding EXIF tag: NULL if not found

See also

gaiaGetExifTags, gaiaExifTagsFree

5.3.3.23 GAIAEXIF_DECLARE gaiaExifTagPtr gaiaGetExifTagByName (const gaiaExifTagListPtr *tag_list*, const char * *tag_name*)

Retrieves an EXIF tag by its name.

Parameters

tag_list	pointer to an EXIF tag list.
tag_name	the Tag Name to be found

Returns

a pointer to the corresponding EXIF tag: NULL if not found

See also

gaiaGetExifTags, gaiaExifTagsFree

5.3.3.24 GAIAEXIF_DECLARE gaiaExifTagPtr gaiaGetExifTagByPos (gaiaExifTagListPtr *tag_list*, const int *pos*)

Retrieves an EXIF tag by its relative position into the list.

Parameters

tag_list	pointer to an EXIF tag list.
pos	relative item position [first item is 0]

Returns

a pointer to the corresponding EXIF tag: NULL if not found

See also

gaiaGetExifTags, gaiaExifTagsFree, gaiaExifTagsCount

5.3.3.25 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

blob	the BLOB to be parsed
size	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.

5.3.3.26 GAIAEXIF_DECLARE int gaiaGetExifTagsCount (gaiaExifTagListPtr *tag_list*)

Return the total number of EXIF tags into the list.

Parameters

tag_list	pointer to an EXIF tag list.

Returns

the EXIF tag count.

See also

gaiaGetExifTags, gaiaExifTagsFree

5.3.3.27 GAIAEXIF_DECLARE int gaiaGetGpsCoords (const unsigned char * *blob*, int *size*, double * *longitude*, double * *latitude*)

Return longitude and latitude from an EXIF-GPS tag.

Parameters

blob	the BLOB to be parsed
size	length of the BLOB (in bytes)
longitude	on success will contain the longitude coordinate
latitude	on success will contain the latitude coordinate

Returns

0 on failure: any other value on success

See also

gaiaGetExifTagById, gaiaGetExifGpsTagById, gaiaGetExifTagByName, gaiaIsExifGpsTag

5.3.3.28 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

blob	the BLOB to be parsed
size	length of the BLOB (in bytes)
latlong	receiving buffer: the text string will be copied here.
ll_size	length of the receiving buffer

Returns

0 on failure: any other value on success

See also

gaiaGetExifTagById, gaiaGetExifGpsTagById, gaiaGetExifTagByName, gaiaIsExifGpsTag

5.3.3.29 GAIAEXIF_DECLARE int gaiaGuessBlobType (const unsigned char * *blob*, int *size*)

Attempts to guess the actual content-type of some BLOB.

Parameters

blob	the BLOB to be parsed
size	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

5.3.3.30 GAIAEXIF_DECLARE int gaiaIsExifGpsTag (const gaiaExifTagPtr *tag*)

Checks if an EXIF tag actually is an EXIF-GPS tag.

Parameters

tag	pointer to an EXIF tag

Returns

0 if false: any other value if true

See also

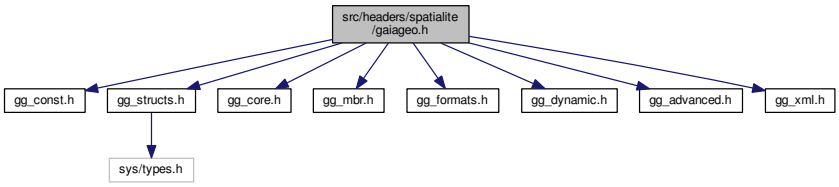
gaiaGetExifTagById, gaiaGetExifGpsTagById, gaiaGetExifTagByName

5.4 src/headers/spatialite/gaiageo.h File Reference

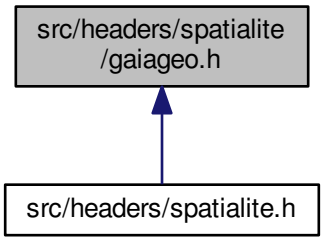
Geometry handling functions and constants.

```
#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:



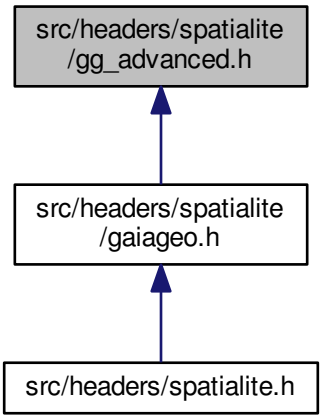
5.4.1 Detailed Description

Geometry handling functions and constants.

5.5 src/headers/spatialite/gg_advanced.h File Reference

Geometry handling functions: advanced.

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 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 (possibile) return a Point Geometry extracted from the latest GEOS error / warning message.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaCriticalPointFromGEOSmsg_r (const void *p_cache)

  Attempts to (possibile) 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 gaiaIsSimple (gaiaGeomCollPtr geom)

  Checks if a Geometry object represents an OGC Simple Geometry.
- GAIAGEO_DECLARE int gaiaIsSimple_r (const void *p_cache, gaiaGeomCollPtr geom)

  Checks if a Geometry object represents an OGC Simple Geometry.
- GAIAGEO_DECLARE int gaiaIsClosed (gaiaLinestringPtr line)

  Checks if a Linestring object represents an OGC Closed Geometry.
- GAIAGEO_DECLARE int gaiaIsClosedGeom (gaiaGeomCollPtr geom)

Checks if a Geometry object represents an OGC Closed Linestring.

- GAIAGEO_DECLARE int gaiaIsClosedGeom_r (const void *p_cache, gaiaGeomCollPtr geom)

  Checks if a Geometry object represents an OGC Closed Linestring.
- GAIAGEO_DECLARE int gaiaIsRing (gaiaLinestringPtr line)

  Checks if a Linestring object represents an OGC Ring Geometry.
- GAIAGEO_DECLARE int gaiaIsRing_r (const void *p_cache, gaiaLinestringPtr line)

  Checks if a Linestring object represents an OGC Ring Geometry.
- GAIAGEO_DECLARE int gaiaIsValid (gaiaGeomCollPtr geom)

  Checks if a Geometry object represents an OGC Valid Geometry.
- GAIAGEO_DECLARE char * gaiaIsValidReason (gaiaGeomCollPtr geom)

  return a TEXT string stating if a Geometry is valid and if not valid, a reason why
- GAIAGEO_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
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaIsValidDetail (gaiaGeomCollPtr geom)

  return a Geometry detail causing a Geometry to be invalid
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaIsValidDetail_r (const void *p_cache, gaiaGeomCollPtr geom)

  return a Geometry detail causing a Geometry to be invalid
- GAIAGEO_DECLARE int gaiaIsValid_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 evalution: Equals.
- GAIAGEO_DECLARE int gaiaGeomCollEquals_r (const void *p_cache, gaiaGeomCollPtr geom1, gaiaGeomCollPtr geom2)

  Spatial relationship evalution: Equals.
- GAIAGEO_DECLARE int gaiaGeomCollDisjoint (gaiaGeomCollPtr geom1, gaiaGeomCollPtr geom2)

  Spatial relationship evalution: Disjoint.
- GAIAGEO_DECLARE int gaiaGeomCollDisjoint_r (const void *p_cache, gaiaGeomCollPtr geom1, gaiaGeomCollPtr geom2)

  Spatial relationship evalution: 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 evalution: Disjoint (GEOSPreparedGeometry)

- GAIAGEO_DECLARE int gaiaGeomCollIntersects (gaiaGeomCollPtr geom1, gaiaGeomCollPtr geom2)

  Spatial relationship evalution: Intesects.
- GAIAGEO_DECLARE int gaiaGeomCollIntersects_r (const void *p_cache, gaiaGeomCollPtr geom1, gaiaGeomCollPtr geom2)

  Spatial relationship evalution: 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 evalution: Intersects (GEOSPreparedGeometry)
- GAIAGEO_DECLARE int gaiaGeomCollOverlaps (gaiaGeomCollPtr geom1, gaiaGeomCollPtr geom2)

  Spatial relationship evalution: Overlaps.
- GAIAGEO_DECLARE int gaiaGeomCollOverlaps_r (const void *p_cache, gaiaGeomCollPtr geom1, gaiaGeomCollPtr geom2)

  Spatial relationship evalution: 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 evalution: Overlaps (GEOSPreparedGeometry)
- GAIAGEO_DECLARE int gaiaGeomCollCrosses (gaiaGeomCollPtr geom1, gaiaGeomCollPtr geom2)

  Spatial relationship evalution: Crosses.
- GAIAGEO_DECLARE int gaiaGeomCollCrosses_r (const void *p_cache, gaiaGeomCollPtr geom1, gaiaGeomCollPtr geom2)

  Spatial relationship evalution: 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 evalution: Crosses (GEOSPreparedGeometry)
- GAIAGEO_DECLARE int gaiaGeomCollContains (gaiaGeomCollPtr geom1, gaiaGeomCollPtr geom2)

  Spatial relationship evalution: Contains.
- GAIAGEO_DECLARE int gaiaGeomCollContains_r (const void *p_cache, gaiaGeomCollPtr geom1, gaiaGeomCollPtr geom2)

  Spatial relationship evalution: 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 evalution: Contains (GEOSPreparedGeometry)
- GAIAGEO_DECLARE int gaiaGeomCollWithin (gaiaGeomCollPtr geom1, gaiaGeomCollPtr geom2)

  Spatial relationship evalution: Within.
- GAIAGEO_DECLARE int gaiaGeomCollWithin_r (const void *p_cache, gaiaGeomCollPtr geom1, gaiaGeomCollPtr geom2)

  Spatial relationship evalution: 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 evalution: Within (GEOSPreparedGeometry)
- GAIAGEO_DECLARE int gaiaGeomCollTouches (gaiaGeomCollPtr geom1, gaiaGeomCollPtr geom2)

  Spatial relationship evalution: Touches.
- GAIAGEO_DECLARE int gaiaGeomCollTouches_r (const void *p_cache, gaiaGeomCollPtr geom1, gaiaGeomCollPtr geom2)

  Spatial relationship evalution: 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 evalution: Touches (GEOSPreparedGeometry)
- GAIAGEO_DECLARE int gaiaGeomCollRelate (gaiaGeomCollPtr geom1, gaiaGeomCollPtr geom2, const char *pattern)

  Spatial relationship evalution: Relate.
- GAIAGEO_DECLARE int gaiaGeomCollRelate_r (const void *p_cache, gaiaGeomCollPtr geom1, gaiaGeomCollPtr geom2, const char *pattern)

Spatial relationship evalution: Relate.

- 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 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 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 ln_geom, double fraction)

  Spatial operator: Line Interpolate Point.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLineInterpolatePoint_r (const void *p_cache, gaiaGeomCollPtr ln_geom, double fraction)

  Spatial operator: Line Interpolate Point.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLineInterpolateEquidistantPoints (gaiaGeomCollPtr ln_geom, double distance)

  Spatial operator: Line Interpolate Equidistant Points.

- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLineInterpolateEquidistantPoints_r (const void *p_cache, gaiaGeomCollPtr ln_geom, double distance)

  Spatial operator: Line Interpolate Equidistant Points.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLineSubstring (gaiaGeomCollPtr ln_geom, double start_fraction, double end_fraction)

  Spatial operator: Line Substring.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLineSubstring_r (const void *p_cache, gaiaGeomCollPtr ln_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 only_edges)

  Utility function: SquareGrid.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSquareGrid_r (const void *p_cache, gaiaGeomCollPtr geom, double origin_x, double origin_y, double size, int only_edges)

  Utility function: SquareGrid.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaTriangularGrid (gaiaGeomCollPtr geom, double origin_x, double origin_y, double size, int only_edges)

  Utility function: TriangularGrid.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaTriangularGrid_r (const void *p_cache, gaiaGeomCollPtr geom, double origin_x, double origin_y, double size, int only_edges)

  Utility function: TriangularGrid.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaHexagonalGrid (gaiaGeomCollPtr geom, double origin_x, double origin_y, double size, int only_edges)

  Utility function: HexagonalGrid.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaHexagonalGrid_r (const void *p_cache, gaiaGeomCollPtr geom, double origin_x, double origin_y, double size, int only_edges)

  Utility function: HexagonalGrid.
- 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 gaiaVoronojDiagram (gaiaGeomCollPtr geom, double extra_frame_size, double tolerance, int only_edges)

  Voronoj Diagram.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaVoronojDiagram_r (const void *p_cache, gaiaGeomCollPtr geom, double extra_frame_size, double tolerance, int only_edges)

  Voronoj 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 gaiaResetLwGeomMsg (void)

  Resets the LWGEOM error and warning messages to an empty state.
- GAIAGEO_DECLARE const char * gaiaGetLwGeomErrorMsg (void)

  Return the latest LWGEOM error message (if any)
- GAIAGEO_DECLARE const char * gaiaGetLwGeomWarningMsg (void)

  Return the latest LWGEOM warning message (if any)
- GAIAGEO_DECLARE void gaiaSetLwGeomErrorMsg (const char *msg)

  Set the current LWGEOM error message.
- GAIAGEO_DECLARE void gaiaSetLwGeomWarningMsg (const char *msg)

  Set the current LWGEOM warning message.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaMakeValid (gaiaGeomCollPtr geom)

  Utility function: MakeValid.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaMakeValidDiscarded (gaiaGeomCollPtr geom)

  Utility function: MakeValidDiscarded.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSegmentize (gaiaGeomCollPtr geom, double dist)

Utility function: Segmentize.

- GAIAGEO_DECLARE int gaiaAzimuth (double xa, double ya, double xb, double yb, double *azimuth)

  Utility function: Azimuth.
- GAIAGEO_DECLARE int gaiaEllipsoidAzimuth (double xa, double ya, double xb, double yb, double a, double b, double *azimuth)

  Utility function: EllipsoidAzimuth.
- GAIAGEO_DECLARE int gaiaProjectedPoint (double x1, double y1, double a, double b, double distance, double azimuth, double *x2, double *y2)

  Utility function: ProjectedPoint.
- GAIAGEO_DECLARE char * gaiaGeoHash (gaiaGeomCollPtr geom, int precision)

  Utility function: GeoHash.
- GAIAGEO_DECLARE char * gaiaAsX3D (gaiaGeomCollPtr geom, const char *srs, int precision, int options, const char *refid)

  Utility function: AsX3D.
- GAIAGEO_DECLARE int gaia3DDistance (gaiaGeomCollPtr geom1, gaiaGeomCollPtr geom2, double *dist)

  Calculates the minimum 3D distance intercurring between two Geometry objects.
- GAIAGEO_DECLARE int gaiaMaxDistance (gaiaGeomCollPtr geom1, gaiaGeomCollPtr geom2, double *dist)

  Calculates the maximum 2D distance intercurring between two Geometry objects.
- GAIAGEO_DECLARE int gaia3DMaxDistance (gaiaGeomCollPtr geom1, gaiaGeomCollPtr geom2, double *dist)

  Calculates the maximum 3D distance intercurring between two Geometry objects.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSplit (gaiaGeomCollPtr input, gaiaGeomCollPtr blade)

  Utility function: Split.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSplitLeft (gaiaGeomCollPtr input, gaiaGeomCollPtr blade)

  Utility function: SplitLeft.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSplitRight (gaiaGeomCollPtr input, gaiaGeomCollPtr blade)

  Utility function: SplitRight.
- GAIAGEO_DECLARE int gaiaGeodesicArea (gaiaGeomCollPtr geom, double a, double b, int use_ellipsoid, double *area)

  Measures the total Area for a Geometry object (geodesic)
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaNodeLines (gaiaGeomCollPtr input)

  Utility function: re-noding lines.
- 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.

5.5.1 Detailed Description

Geometry handling functions: advanced.

5.5.2 Function Documentation

5.5.2.1 GAIAGEO_DECLARE int gaia3DDistance (gaiaGeomCollPtr *geom1,* gaiaGeomCollPtr *geom2,* double * *dist*)

Calculates the minimum 3D distance intercurring between two Geometry objects.

Parameters

geom1	the first Geometry object
geom2	the second Geometry object
dist	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

LWGEOM support required.

5.5.2.2 GAIAGEO_DECLARE int gaia3DMaxDistance (gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*, double * *dist*)

Calculates the maximum 3D distance intercurring between two Geometry objects.

Parameters

geom1	the first Geometry object
geom2	the second Geometry object
dist	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

LWGEOM support required.

5.5.2.3 GAIAGEO_DECLARE char* gaiaAsX3D (gaiaGeomCollPtr *geom*, const char * *srs*, int *precision*, int *options*, const char * *refid*)

Utility function: AsX3D.

Parameters

geom	the input geometry.
srs	the WKT SRS definition.
precision	the expected precision (coord decimal digits).
options	
refid	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

LWGEOM support required.

5.5.2.4 GAIAGEO_DECLARE int gaiaAzimuth (double *xa*, double *ya*, double *xb*, double *yb*, double * *azimuth*)

Utility function: Azimuth.

Parameters

xa	the X coordinate of PointA.
ya	the Y coordinate of PointA.
xb	the X ccordinate of PointB.
yb	the Y coordinate of PointB.
azimuth	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

LWGEOM support required.

5.5.2.5 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaBoundary (gaiaGeomCollPtr *geom*)

Spatial operator: Boundary.

Parameters

geom	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.

5.5.2.6 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaBoundary_r (const void * *p_cache,* gaiaGeomCollPtr *geom*)

Spatial operator: Boundary.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	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.

5.5.2.7 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaConcaveHull (gaiaGeomCollPtr *geom,* double *factor,* double *tolerance,* int *allow_holes*)

Concave Hull.

Parameters

geom	pointer to input Geometry object.
factor	multiplier used for filtering Delaunay triangles: please read the note.
tolerance	optional snapping tolerance.
allow_holes	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

Note

This function will first create the Delauany 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.

5.5.2.8 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaConcaveHull_r (const void ∗ *p_cache*, gaiaGeomCollPtr *geom*, double *factor*, double *tolerance*, int *allow_holes*)

Concave Hull.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	pointer to input Geometry object.
factor	multiplier used for filtering Delaunay triangles: please read the note.
tolerance	optional snapping tolerance.
allow_holes	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

Note

This function will first create the Delauany 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.

5.5.2.9 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaConvexHull (gaiaGeomCollPtr *geom*)

Spatial operator: ConvexHull.

Parameters

geom	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.

5.5.2.10 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaConvexHull_r (const void ∗ *p_cache*, gaiaGeomCollPtr *geom*)

Spatial operator: ConvexHull.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	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.

5.5.2.11 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaCriticalPointFromGEOSmsg (void)

Attempts to (possibile) return a Point Geometry extracted from the latest GEOS error / warning message.

Returns

a Point Geometry: NULL if no warning/error was previoysly 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.

5.5.2.12 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaCriticalPointFromGEOSmsg_r (const void * *p_cache*)

Attempts to (possibile) return a Point Geometry extracted from the latest GEOS error / warning message.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()

Returns

a Point Geometry: NULL if no warning/error was previoysly 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.

5.5.2.13 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaDelaunayTriangulation (gaiaGeomCollPtr *geom,* double *tolerance,* int *only_edges*)

Delaunay Triangulation.

Parameters

geom	pointer to input Geometry object.
tolerance	optional snapping tolerance.
only_edges	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

gaiaDelaunatTriangulation_r, gaiaFreeGeomColl, gaiaVoronojDiagram, 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.

5.5.2.14 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaDelaunayTriangulation_r (const void * *p_cache*, gaiaGeomCollPtr *geom*, double *tolerance*, int *only_edges*)

Delaunay Triangulation.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	pointer to input Geometry object.
tolerance	optional snapping tolerance.
only_edges	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

gaiaDelaunatTriangulation, gaiaFreeGeomColl, gaiaVoronojDiagram, 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.

5.5.2.15 GAIAGEO_DECLARE int gaiaEllipsoidAzimuth (double *xa*, double *ya*, double *xb*, double *yb*, double *a*, double *b*, double * *azimuth*)

Utility function: EllipsoidAzimuth.

Parameters

xa	the X coordinate of PointA.
ya	the Y coordinate of PointA.
xb	the X ccordinate of PointB.
yb	the Y coordinate of PointB.
a	major axis of the reference spheroid.
b	minor axis of the reference spheroid.
azimuth	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

LWGEOM support required.

5.5.2.16 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaFromGeos_XY (const void * *geos*)

Converts a GEOS Geometry into a Geometry object [XY dims].

Parameters

geos	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 usafe.

Remarks

GEOS support required.

5.5.2.17 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaFromGeos_XY_r (const void * *p_cache,* const void * *geos*)

Converts a GEOS Geometry into a Geometry object [XY dims].

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geos	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.

5.5.2.18 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaFromGeos_XYM (const void * *geos*)

Converts a GEOS Geometry into a Geometry object [XYM dims].

Parameters

geos	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.

5.5.2.19 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaFromGeos_XYM_r (const void * *p_cache*, const void * *geos*)

Converts a GEOS Geometry into a Geometry object [XYM dims].

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geos	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.

5.5.2.20 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaFromGeos_XYZ (const void * *geos*)

Converts a GEOS Geometry into a Geometry object [XYZ dims].

Parameters

geos	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.

5.5.2.21 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaFromGeos_XYZ_r (const void * *p_cache*, const void * *geos*)

Converts a GEOS Geometry into a Geometry object [XYZ dims].

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geos	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.

5.5.2.22 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaFromGeos_XYZM (const void * *geos*)

Converts a GEOS Geometry into a Geometry object [XYZM dims].

Parameters

geos	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.

5.5.2.23 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaFromGeos_XYZM_r (const void * *p_cache,* const void * *geos*)

Converts a GEOS Geometry into a Geometry object [XYZM dims].

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geos	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.

5.5.2.24 GAIAGEO_DECLARE int gaiaGeodesicArea (gaiaGeomCollPtr *geom*, double *a*, double *b*, int *use_ellipsoid*, double * *area*)

Measures the total Area for a Geometry object (geodesic)

Parameters

geom	pointer to Geometry object
a	major axis of the reference spheroid.
b	minor axis of the reference spheroid.
use_ellipsoid	if TRUE will measure the Area on the Ellipsoid, otherwise on the Sphere
area	on completion this variable will contain the measured area

Returns

0 on failure: any other value on success

See also

gaiaGeomCollLength, gaiaMeasureArea, gaiaGeomCollArea

Remarks

LWGEOM support required.

5.5.2.25 GAIAGEO_DECLARE char* gaiaGeoHash (gaiaGeomCollPtr *geom*, int *precision*)

Utility function: GeoHash.

Parameters

geom	the input geometry.
precision	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

LWGEOM support required.

5.5.2.26 GAIAGEO_DECLARE int gaiaGeomCollArea (gaiaGeomCollPtr *geom*, double * *area*)

Measures the total Area for a Geometry object.

Parameters

geom	pointer to Geometry object
area	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.

5.5.2.27 GAIAGEO_DECLARE int gaiaGeomCollArea_r (const void * *p_cache*, gaiaGeomCollPtr *geom*, double * *area*)

Measures the total Area for a Geometry object.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	pointer to Geometry object
area	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.

5.5.2.28 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeomCollBuffer (gaiaGeomCollPtr *geom*, double *radius*, int *points*)

Spatial operator: Buffer.

Parameters

geom	the input Geometry object
radius	the buffer's radius
points	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.

5.5.2.29 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeomCollBuffer_r (const void * *p_cache*, gaiaGeomCollPtr *geom*, double *radius*, int *points*)

Spatial operator: Buffer.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	the input Geometry object
radius	the buffer's radius
points	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.

5.5.2.30 GAIAGEO_DECLARE int gaiaGeomCollCentroid (gaiaGeomCollPtr *geom*, double * *x*, double * *y*)

Spatial operator: Centroid.

Parameters

geom	pointer to Geometry object.
x	on completion this variable will contain the centroid X coordinate
y	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.

5.5.2.31 GAIAGEO_DECLARE int gaiaGeomCollCentroid_r (const void * *p_cache*, gaiaGeomCollPtr *geom*, double * *x*, double * *y*)

Spatial operator: Centroid.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	pointer to Geometry object.
x	on completion this variable will contain the centroid X coordinate
y	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.

5.5.2.32 GAIAGEO_DECLARE int gaiaGeomCollContains (gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial relationship evalution: Contains.

Parameters

geom1	the first Geometry object to be evaluated
geom2	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.

5.5.2.33 GAIAGEO_DECLARE int gaiaGeomCollContains_r (const void * *p_cache*, gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial relationship evalution: Contains.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	the first Geometry object to be evaluated
geom2	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, gaiaGeom↩CollTouches, gaiaGeomCollRelate

Note

reentrant and thread-safe.

Remarks

GEOS support required.

5.5.2.34 GAIAGEO_DECLARE int gaiaGeomCollCoveredBy (gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Topology check: test if a Geometry is covered by another one.

Parameters

geom1	pointer to first input Geometry object.
geom2	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.

5.5.2.35 GAIAGEO_DECLARE int gaiaGeomCollCoveredBy_r (const void * *p_cache*, gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Topology check: test if a Geometry is covered by another one.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	pointer to first input Geometry object.
geom2	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.

5.5.2.36 GAIAGEO_DECLARE int gaiaGeomCollCovers (gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Topology check: test if a Geometry covers another one.

Parameters

geom1	pointer to first input Geometry object.
geom2	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.

5.5.2.37 GAIAGEO_DECLARE int gaiaGeomCollCovers_r (const void * *p_cache*, gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Topology check: test if a Geometry covers another one.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	pointer to first input Geometry object.
geom2	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.

5.5.2.38 GAIAGEO_DECLARE int gaiaGeomCollCrosses (gaiaGeomCollPtr *geom1,* gaiaGeomCollPtr *geom2*)

Spatial relationship evalution: Crosses.

Parameters

geom1	the first Geometry object to be evaluated
geom2	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.

5.5.2.39 GAIAGEO_DECLARE int gaiaGeomCollCrosses_r (const void * *p_cache,* gaiaGeomCollPtr *geom1,* gaiaGeomCollPtr *geom2*)

Spatial relationship evalution: Crosses.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	the first Geometry object to be evaluated
geom2	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.

5.5.2.40 GAIAGEO_DECLARE int gaiaGeomCollDisjoint (gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial relationship evalution: Disjoint.

Parameters

geom1	the first Geometry object to be evaluated
geom2	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.

5.5.2.41 GAIAGEO_DECLARE int gaiaGeomCollDisjoint_r (const void * *p_cache*, gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial relationship evalution: Disjoint.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	the first Geometry object to be evaluated
geom2	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.

5.5.2.42 GAIAGEO_DECLARE int gaiaGeomCollDistance (gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*, double * *dist*)

Calculates the maximum distance intercurring between two Geometry objects.

Parameters

geom1	the first Geometry object
geom2	the second Geometry object
dist	on completion this variable will contain the calculated distance

Returns

0 on failure: any other value on success.

See also

gaiaGeomCollDistance_r, gaia3DDistance, gaiaMaxDistance, gaia3DMaxDistance

Note

this function always computes the 2D cartesian distance.
not reentrant and thread unsafe.

Remarks

GEOS support required.

5.5.2.43 GAIAGEO_DECLARE int gaiaGeomCollDistance_r (const void * *p_cache*, gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*, double * *dist*)

Calculates the maximum distance intercurring between two Geometry objects.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	the first Geometry object
geom2	the second Geometry object
dist	on completion this variable will contain the calculated distance

Returns

0 on failure: any other value on success.

See also

gaiaGeomCollDistance, gaia3DDistance, gaiaMaxDistance, gaia3DMaxDistance

Note

this function always computes the 2D cartesian distance.
reentrant and thread-safe.

Remarks

GEOS support required.

5.5.2.44 GAIAGEO_DECLARE int gaiaGeomCollEquals (gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial relationship evalution: Equals.

Parameters

geom1	the first Geometry object to be evaluated
geom2	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.

5.5.2.45 GAIAGEO_DECLARE int gaiaGeomCollEquals_r (const void * *p_cache*, gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial relationship evalution: Equals.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	the first Geometry object to be evaluated
geom2	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.

5.5.2.46 GAIAGEO_DECLARE int gaiaGeomCollIntersects (gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial relationship evalution: Intesects.

Parameters

geom1	the first Geometry object to be evaluated
geom2	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.

5.5.2.47 GAIAGEO_DECLARE int gaiaGeomCollIntersects_r (const void * *p_cache*, gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial relationship evalution: Intersects.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	the first Geometry object to be evaluated
geom2	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.

5.5.2.48 GAIAGEO_DECLARE int gaiaGeomCollLength (gaiaGeomCollPtr *geom*, double * *length*)

Measures the total Length for a Geometry object.

Parameters

geom	pointer to Geometry object
length	on completion this variable will contain the measured length

Returns

0 on failure: any other value on success

See also

gaiaGeomCollLenght_r, gaiaGeomCollArea, gaiaMeasureLength, gaiaGeomCollLengthOrPerimeter

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

Examples:

demo1.c.

5.5.2.49 GAIAGEO_DECLARE int gaiaGeomCollLength_r (const void * *p_cache*, gaiaGeomCollPtr *geom*, double * *length*)

Measures the total Length for a Geometry object.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	pointer to Geometry object
length	on completion this variable will contain the measured length

Returns

0 on failure: any other value on success

See also

gaiaGeomCollLenght, gaiaGeomCollArea, gaiaMeasureLength, gaiaGeomCollLengthOrPerimeter

Note

reentrant and thread-safe.

Remarks

GEOS support required.

5.5.2.50 GAIAGEO_DECLARE int gaiaGeomCollLengthOrPerimeter (gaiaGeomCollPtr *geom*, int *perimeter*, double * *length*)

Measures the total Length or Perimeter for a Geometry object.

Parameters

geom	pointer to Geometry object
perimeter	if TRUE only Polygons will be considered, ignoring any Linesting the opposite if FALSE (considering only Linestrings and ignoring any Polygon)
length	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.

5.5.2.51 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.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	pointer to Geometry object
perimeter	if TRUE only Polygons will be considered, ignoring any Linesting the opposite if FALSE (considering only Linestrings and ignoring any Polygon)
length	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.

5.5.2.52 GAIAGEO_DECLARE int gaiaGeomCollOverlaps (gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial relationship evalution: Overlaps.

Parameters

geom1	the first Geometry object to be evaluated
geom2	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.

5.5.2.53 GAIAGEO_DECLARE int gaiaGeomCollOverlaps_r (const void * *p_cache*, gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial relationship evalution: Overlaps.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	the first Geometry object to be evaluated
geom2	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.

5.5.2.54 GAIAGEO_DECLARE int gaiaGeomCollPreparedContains (const void * *p_cache*, gaiaGeomCollPtr *geom1*, unsigned char * *blob1*, int *size1*, gaiaGeomCollPtr *geom2*, unsigned char * *blob2*, int *size2*)

Spatial relationship evalution: Contains (GEOSPreparedGeometry)

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	the first Geometry object to be evaluated
blob1	the BLOB corresponding to the first Geometry
size1	the size (in bytes) of the first BLOB
geom2	the second Geometry object to be evaluated
blob2	the BLOB corresponding to the second Geometry
size2	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.

5.5.2.55 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

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	pointer to first input Geometry object.
blob1	the BLOB corresponding to the first Geometry
size1	the size (in bytes) of the first BLOB
geom2	pointer to second input Geometry object.
blob2	the BLOB corresponding to the second Geometry
size2	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.

5.5.2.56 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)

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	pointer to first input Geometry object.
blob1	the BLOB corresponding to the first Geometry
size1	the size (in bytes) of the first BLOB
geom2	pointer to second input Geometry object.
blob2	the BLOB corresponding to the second Geometry
size2	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.

5.5.2.57 GAIAGEO_DECLARE int gaiaGeomCollPreparedCrosses (const void * *p_cache*, gaiaGeomCollPtr *geom1*, unsigned char * *blob1*, int *size1*, gaiaGeomCollPtr *geom2*, unsigned char * *blob2*, int *size2*)

Spatial relationship evalution: Crosses (GEOSPreparedGeometry)

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	the first Geometry object to be evaluated
blob1	the BLOB corresponding to the first Geometry
size1	the size (in bytes) of the first BLOB
geom2	the second Geometry object to be evaluated
blob2	the BLOB corresponding to the second Geometry
size2	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.

5.5.2.58 GAIAGEO_DECLARE int gaiaGeomCollPreparedDisjoint (const void * *p_cache*, gaiaGeomCollPtr *geom1*, unsigned char * *blob1*, int *size1*, gaiaGeomCollPtr *geom2*, unsigned char * *blob2*, int *size2*)

Spatial relationship evalution: Disjoint (GEOSPreparedGeometry)

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	the first Geometry object to be evaluated
blob1	the BLOB corresponding to the first Geometry
size1	the size (in bytes) of the first BLOB
geom2	the second Geometry object to be evaluated
blob2	the BLOB corresponding to the second Geometry
size2	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.

5.5.2.59 GAIAGEO_DECLARE int gaiaGeomCollPreparedIntersects (const void * *p_cache,* gaiaGeomCollPtr *geom1,* unsigned char * *blob1,* int *size1,* gaiaGeomCollPtr *geom2,* unsigned char * *blob2,* int *size2*)

Spatial relationship evalution: Intersects (GEOSPreparedGeometry)

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	the first Geometry object to be evaluated
blob1	the BLOB corresponding to the first Geometry
size1	the size (in bytes) of the first BLOB
geom2	the second Geometry object to be evaluated
blob2	the BLOB corresponding to the second Geometry
size2	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.

5.5.2.60 GAIAGEO_DECLARE int gaiaGeomCollPreparedOverlaps (const void * *p_cache*, gaiaGeomCollPtr *geom1*, unsigned char * *blob1*, int *size1*, gaiaGeomCollPtr *geom2*, unsigned char * *blob2*, int *size2*)

Spatial relationship evalution: Overlaps (GEOSPreparedGeometry)

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	the first Geometry object to be evaluated
blob1	the BLOB corresponding to the first Geometry
size1	the size (in bytes) of the first BLOB
geom2	the second Geometry object to be evaluated
blob2	the BLOB corresponding to the second Geometry
size2	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.

5.5.2.61 GAIAGEO_DECLARE int gaiaGeomCollPreparedTouches (const void * *p_cache*, gaiaGeomCollPtr *geom1*, unsigned char * *blob1*, int *size1*, gaiaGeomCollPtr *geom2*, unsigned char * *blob2*, int *size2*)

Spatial relationship evalution: Touches (GEOSPreparedGeometry)

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	the first Geometry object to be evaluated
blob1	the BLOB corresponding to the first Geometry
size1	the size (in bytes) of the first BLOB
geom2	the second Geometry object to be evaluated
blob2	the BLOB corresponding to the second Geometry
size2	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.

5.5.2.62 GAIAGEO_DECLARE int gaiaGeomCollPreparedWithin (const void * *p_cache*, gaiaGeomCollPtr *geom1*, unsigned char * *blob1*, int *size1*, gaiaGeomCollPtr *geom2*, unsigned char * *blob2*, int *size2*)

Spatial relationship evalution: Within (GEOSPreparedGeometry)

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	the first Geometry object to be evaluated
blob1	the BLOB corresponding to the first Geometry
size1	the size (in bytes) of the first BLOB
geom2	the second Geometry object to be evaluated
blob2	the BLOB corresponding to the second Geometry
size2	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.

5.5.2.63 GAIAGEO_DECLARE int gaiaGeomCollRelate (gaiaGeomCollPtr *geom1,* gaiaGeomCollPtr *geom2,* const char * *pattern*)

Spatial relationship evalution: Relate.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	the first Geometry object to be evaluated
geom2	the second Geometry object to be evaluated
pattern	intersection matrix pattern [DE-9IM]

Returns

0 if false: any other value if true

See also

gaiaGeomCollRelate_r, gaiaGeomCollEquals, gaiaGeomCollDisjoint, gaiaGeomCollIntersects, gaiaGeomCollOverlaps, gaiaGeomCollCrosses, gaiaGeomCollContains, gaiaGeomCollWithin, gaiaGeomCollRelate

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

5.5.2.64 GAIAGEO_DECLARE int gaiaGeomCollRelate_r (const void * *p_cache*, gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*, const char * *pattern*)

Spatial relationship evalution: Relate.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	the first Geometry object to be evaluated
geom2	the second Geometry object to be evaluated
pattern	intersection matrix pattern [DE-9IM]

Returns

0 if false: any other value if true

See also

gaiaGeomCollRelate, gaiaGeomCollEquals, gaiaGeomCollDisjoint, gaiaGeomCollIntersects, gaiaGeomCollOverlaps, gaiaGeomCollCrosses, gaiaGeomCollContains, gaiaGeomCollWithin, gaiaGeomCollRelate

Note

reentrant and thread-safe.

Remarks

GEOS support required.

5.5.2.65 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeomCollSimplify (gaiaGeomCollPtr *geom*, double *tolerance*)

Spatial operator: Simplify.

Parameters

geom	the input Geometry object
tolerance	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.

5.5.2.66 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeomCollSimplify_r (const void * *p_cache*, gaiaGeomCollPtr *geom*, double *tolerance*)

Spatial operator: Simplify.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	the input Geometry object
tolerance	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.

5.5.2.67 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeomCollSimplifyPreserveTopology (gaiaGeomCollPtr *geom*, double *tolerance*)

Spatial operator: Simplify [preserving topology].

Parameters

geom	the input Geometry object
tolerance	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.

5.5.2.68 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeomCollSimplifyPreserveTopology_r (const void * *p_cache*, gaiaGeomCollPtr *geom*, double *tolerance*)

Spatial operator: Simplify [preserving topology].

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	the input Geometry object
tolerance	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.

5.5.2.69 GAIAGEO_DECLARE int gaiaGeomCollTouches (gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial relationship evalution: Touches.

Parameters

geom1	the first Geometry object to be evaluated
geom2	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.

5.5.2.70 GAIAGEO_DECLARE int gaiaGeomCollTouches_r (const void * *p_cache*, gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial relationship evalution: Touches.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	the first Geometry object to be evaluated
geom2	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.

5.5.2.71 GAIAGEO_DECLARE int gaiaGeomCollWithin (gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial relationship evalution: Within.

Parameters

geom1	the first Geometry object to be evaluated
geom2	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.

5.5.2.72 GAIAGEO_DECLARE int gaiaGeomCollWithin_r (const void * *p_cache*, gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial relationship evalution: Within.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	the first Geometry object to be evaluated
geom2	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.

5.5.2.73 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeometryDifference (gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial operator: Difference.

Parameters

geom1	the first Geometry object
geom2	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.

5.5.2.74 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeometryDifference_r (const void * *p_cache*, gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial operator: Difference.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	the first Geometry object
geom2	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.

5.5.2.75 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeometryIntersection (gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial operator: Intersection.

Parameters

geom1	the first Geometry object
geom2	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, gaia↩GeometrySymDifference, 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.

5.5.2.76 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeometryIntersection_r (const void * *p_cache,* gaiaGeomCollPtr *geom1,* gaiaGeomCollPtr *geom2*)

Spatial operator: Intersection.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	the first Geometry object
geom2	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, gaia↩GeometrySymDifference, 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.

5.5.2.77 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeometrySymDifference (gaiaGeomCollPtr *geom1,* gaiaGeomCollPtr *geom2*)

Spatial operator: SymDifference.

Parameters

geom1	the first Geometry object
geom2	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.

5.5.2.78 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeometrySymDifference_r (const void * *p_cache*, gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial operator: SymDifference.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	the first Geometry object
geom2	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.

5.5.2.79 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeometryUnion (gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial operator: Union.

Parameters

geom1	the first Geometry object
geom2	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.

5.5.2.80 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaGeometryUnion_r (const void * *p_cache*, gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial operator: Union.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	the first Geometry object
geom2	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.

5.5.2.81 GAIAGEO_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 previoysly found.

See also

gaiaGetGeosAuxErrorMsg_r, gaiaResetGeosMsg, gaiaGetGeosErrorMsg, gaiaGetGeosWarningMsg, gaiaSetGeosErrorMsg, gaiaSetGeosWarningMsg, gaiaSetGeosAuxErrorMsg

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

5.5.2.82 GAIAGEO_DECLARE const char* gaiaGetGeosAuxErrorMsg_r (const void * *p_cache*)

Return the latest GEOS (auxiliary) error message (if any)

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()

Returns

the latest GEOS (auxiliary) error message: an empty string if no error was previoysly found.

See also

gaiaGetGeosAuxErrorMsg, gaiaResetGeosMsg, gaiaGetGeosErrorMsg, gaiaGetGeosWarningMsg, gaiaSetGeosErrorMsg, gaiaSetGeosWarningMsg, gaiaSetGeosAuxErrorMsg

Note

reentrant and thread-safe.

Remarks

GEOS support required.

5.5.2.83 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 previoysly found.

See also

gaiaGetGeosErrorMsg_r, gaiaResetGeosMsg, gaiaGetGeosWarningMsg, gaiaGetGeosAuxErrorMsg, gaiaSetGeosErrorMsg, gaiaSetGeosWarningMsg, gaiaSetGeosAuxErrorMsg, gaiaCriticalPointFromGEOSmsg

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

5.5.2.84 GAIAGEO_DECLARE const char* gaiaGetGeosErrorMsg_r (const void * *p_cache*)

Return the latest GEOS error message (if any)

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()

Returns

the latest GEOS error message: an empty string if no error was previoysly found.

See also

gaiaGetGeosErrorMsg, gaiaResetGeosMsg, gaiaGetGeosWarningMsg, gaiaGetGeosAuxErrorMsg, gaiaSetGeosErrorMsg, gaiaSetGeosWarningMsg, gaiaSetGeosAuxErrorMsg, gaiaCriticalPointFromGEOSmsg

Note

reentrant and thread-safe.

Remarks

GEOS support required.

5.5.2.85 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 previoysly found.

See also

gaiaGetGeosWarningMsg_r, gaiaResetGeosMsg, gaiaGetGeosErrorMsg, gaiaGetGeosAuxErrorMsg, gaiaSetGeosErrorMsg, gaiaSetGeosWarningMsg, gaiaSetGeosAuxErrorMsg, gaiaCriticalPointFromGEOSmsg

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

5.5.2.86 GAIAGEO_DECLARE const char* gaiaGetGeosWarningMsg_r (const void * *p_cache*)

Return the latest GEOS warning message (if any)

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()

Returns

the latest GEOS warning message: an empty string if no warning was previoysly found.

See also

gaiaGetGeosWarningMsg, gaiaResetGeosMsg, gaiaGetGeosErrorMsg, gaiaGetGeosAuxErrorMsg, gaiaSetGeosErrorMsg, gaiaSetGeosWarningMsg, gaiaSetGeosAuxErrorMsg, gaiaCriticalPointFromGEOSmsg

Note

reentrant and thread-safe.

Remarks

GEOS support required.

5.5.2.87 GAIAGEO_DECLARE const char* gaiaGetLwGeomErrorMsg (void)

Return the latest LWGEOM error message (if any)

Returns

the latest LWGEOM error message: an empty string if no error was previoysly found.

Note

not reentrant and thread unsafe.

See also

gaiaResetLwGeomMsg, gaiaGetLwGeomWarningMsg, gaiaSetLwGeomErrorMsg, gaiaSetLwGeomWarningMsg

Remarks

LWGEOM support required.

5.5.2.88 GAIAGEO_DECLARE const char* gaiaGetLwGeomWarningMsg (void)

Return the latest LWGEOM warning message (if any)

Returns

the latest LWGEOM warning message: an empty string if no warning was previoysly found.

See also

gaiaResetLwGeomMsg, gaiaGetLwGeomErrorMsg, gaiaSetLwGeomErrorMsg, gaiaSetLwGeomWarningMsg

Note

not reentrant and thread unsafe.

Remarks

LWGEOM support required.

5.5.2.89 GAIAGEO_DECLARE int gaiaGetPointOnSurface (gaiaGeomCollPtr *geom*, double * *x*, double * *y*)

Spatial operator: PointOnSurface.

Parameters

geom	pointer to Geometry object.
x	on completion this variable will contain the Point X coordinate
y	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.

5.5.2.90 GAIAGEO_DECLARE int gaiaGetPointOnSurface_r (const void * *p_cache*, gaiaGeomCollPtr *geom*, double * *x*, double * *y*)

Spatial operator: PointOnSurface.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	pointer to Geometry object.
x	on completion this variable will contain the Point X coordinate
y	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.

5.5.2.91 GAIAGEO_DECLARE int gaiaHausdorffDistance (gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*, double * *dist*)

Calculates the Hausdorff distance intercurring between two Geometry objects.

Parameters

geom1	pointer to first Geometry object
geom2	pointer to second Geometry object
dist	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.

5.5.2.92 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

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	pointer to first Geometry object
geom2	pointer to second Geometry object
dist	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.

5.5.2.93 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaHexagonalGrid (gaiaGeomCollPtr *geom*, double *origin_x*, double *origin_y*, double *size*, int *only_edges*)

Utility function: HexagonalGrid.

Parameters

geom	the Geometry to be covered by the Grid.
origin_x	the X ccordinate identifying the Grid Origin.
origin_y	the Y coordinate identifiying the Grid Origin.
size	the Grid cell-side size.
only_edges	if non-zero will return a MULTILINESTRING, otherwise it 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.

5.5.2.94 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaHexagonalGrid_r (const void * *p_cache*, gaiaGeomCollPtr *geom*, double *origin_x*, double *origin_y*, double *size*, int *only_edges*)

Utility function: HexagonalGrid.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	the Geometry to be covered by the Grid.
origin_x	the X ccordinate identifying the Grid Origin.
origin_y	the Y coordinate identifiying the Grid Origin.
size	the Grid cell-side size.
only_edges	if non-zero will return a MULTILINESTRING, otherwise it 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.

5.5.2.95 GAIAGEO_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 gaiaIsClosedGeom for an alternative.

Parameters

line	pointer to Linestring object.

Returns

0 if false; any other value if true

See also

gaiaIsSimple, gaiaIsRing, gaiaIsValid, gaiaIsClosedGeom

Remarks

GEOS support required.

5.5.2.96 GAIAGEO_DECLARE int gaiaIsClosedGeom (gaiaGeomCollPtr *geom*)

Checks if a Geometry object represents an OGC Closed Linestring.

Parameters

geom	pointer to Geometry object.

Returns

0 if false; any other value if true

See also

gaiaIsClosedGeom_r, gaiaIsSimple, gaiaIsRing, gaiaIsValid, gaiaIsClosed

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

5.5.2.97 GAIAGEO_DECLARE int gaiaIsClosedGeom_r (const void * *p_cache,* gaiaGeomCollPtr *geom*)

Checks if a Geometry object represents an OGC Closed Linestring.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	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.

5.5.2.98 GAIAGEO_DECLARE int gaiaIsRing (gaiaLinestringPtr *line*)

Checks if a Linestring object represents an OGC Ring Geometry.

Parameters

line	pointer to Geometry object.

Returns

0 if false; any other value if true

See also

gaiaIsRing_r, gaiaIsSimple, gaiaIsClosed, gaiaIsValid

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

5.5.2.99 GAIAGEO_DECLARE int gaiaIsRing_r (const void * *p_cache*, gaiaLinestringPtr *line*)

Checks if a Linestring object represents an OGC Ring Geometry.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
line	pointer to Geometry object.

Returns

0 if false; any other value if true

See also

gaiaIsRing, gaiaIsSimple, gaiaIsClosed, gaiaIsValid

Note

reentrant and thread-safe.

Remarks

GEOS support required.

5.5.2.100 GAIAGEO_DECLARE int gaiaIsSimple (gaiaGeomCollPtr *geom*)

Checks if a Geometry object represents an OGC Simple Geometry.

Parameters

geom	pointer to Geometry object.

Returns

0 if false; any other value if true

See also

gaiaIsSimple_r, gaiaIsClosed, gaiaIsRing, gaiaIsValid

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

5.5.2.101 GAIAGEO_DECLARE int gaiaIsSimple_r (const void * *p_cache*, gaiaGeomCollPtr *geom*)

Checks if a Geometry object represents an OGC Simple Geometry.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()

geom	pointer to Geometry object.

Returns

0 if false; any other value if true

See also

gaiaIsSimple, gaiaIsClosed, gaiaIsRing, gaiaIsValid

Note

reentrant and thread-safe.

Remarks

GEOS support required.

5.5.2.102 GAIAGEO_DECLARE int gaiaIsValid (gaiaGeomCollPtr *geom*)

Checks if a Geometry object represents an OGC Valid Geometry.

Parameters

geom	pointer to Geometry object.

Returns

0 if false; any other value if true

See also

gaiaIsValid_r, gaiaIsSimple, gaiaIsClosed, gaiaIsRing, gaiaIsValidReason

Note

not reentrant and thread unsafe.

Remarks

GEOS support required.

Examples:

demo2.c.

5.5.2.103 GAIAGEO_DECLARE int gaiaIsValid_r (const void * *p_cache*, gaiaGeomCollPtr *geom*)

Checks if a Geometry object represents an OGC Valid Geometry.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	pointer to Geometry object.

Returns

0 if false; any other value if true

See also

gaiaIsValid, gaiaIsSimple, gaiaIsClosed, gaiaIsRing, gaiaIsValidReason_r

Note

reentrant and thread-safe.

Remarks

GEOS support required.

5.5.2.104 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaIsValidDetail (gaiaGeomCollPtr *geom*)

return a Geometry detail causing a Geometry to be invalid

Parameters

geom	pointer to the Geometry object to be validated.

Returns

pointer to a Geometry object causing invalidity, or NULL.

See also

gaiaIsValid, gaiaIsValidReason, gaiaIsValidDetail_r

Note

you are responsible to destroy the returned Geometry
not reentrant and thread unsafe.

Remarks

GEOS support required.

5.5.2.105 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaIsValidDetail_r (const void * *p_cache*, gaiaGeomCollPtr *geom*)

return a Geometry detail causing a Geometry to be invalid

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	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.

5.5.2.106 GAIAGEO_DECLARE char* gaiaIsValidReason (gaiaGeomCollPtr *geom*)

return a TEXT string stating if a Geometry is valid and if not valid, a reason why

Parameters

geom	pointer to the Geometry object to be validated.

Returns

a text string.

See also

gaiaIsValid, gaiaIsValidReason_r, gaiaIsValidDetail

Note

you are responsible to free() the returned text string
not reentrant and thread unsafe.

Remarks

GEOS support required.

5.5.2.107 GAIAGEO_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

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	pointer to the Geometry object to be validated.

Returns

a text string.

See also

gaiaIsValid_r, gaiaIsValidReason, gaiaIsValidDetail_r

Note

you are responsible to free() the returned text string
reentrant and thread-safe.

Remarks

GEOS support required.

5.5.2.108 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLineInterpolateEquidistantPoints (gaiaGeomCollPtr *ln_geom*, double *distance*)

Spatial operator: Line Interpolate Equidistant Points.

Parameters

ln_geom	the input Geometry object [expected to be of lineal type]
distance	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.

5.5.2.109 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLineInterpolateEquidistantPoints_r (const void * *p_cache*, gaiaGeomCollPtr *ln_geom*, double *distance*)

Spatial operator: Line Interpolate Equidistant Points.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
ln_geom	the input Geometry object [expected to be of lineal type]
distance	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.

5.5.2.110 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLineInterpolatePoint (gaiaGeomCollPtr *ln_geom,* double *fraction*)

Spatial operator: Line Interpolate Point.

Parameters

ln_geom	the input Geometry object [expected to be of lineal type]
fraction	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.

5.5.2.111 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLineInterpolatePoint_r (const void * *p_cache,* gaiaGeomCollPtr *ln_geom,* double *fraction*)

Spatial operator: Line Interpolate Point.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
ln_geom	the input Geometry object [expected to be of lineal type]
fraction	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.

5.5.2.112 GAIAGEO_DECLARE double gaiaLineLocatePoint (gaiaGeomCollPtr *ln_geom*, gaiaGeomCollPtr *pt_geom*)

Determines the location of the closest Point on Linestring to the given Point.

Parameters

ln_geom	pointer to first input Geometry object [expected to be of the lineal type].
pt_geom	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.

5.5.2.113 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

p_cache	a memory pointer returned by spatialite_alloc_connection()
ln_geom	pointer to first input Geometry object [expected to be of the lineal type].
pt_geom	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.

5.5.2.114 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLineMerge (gaiaGeomCollPtr *geom*)

Spatial operator: Line Merge.

Parameters

geom	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.

5.5.2.115 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLineMerge_r (const void * *p_cache,* gaiaGeomCollPtr *geom*)

Spatial operator: Line Merge.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	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.

5.5.2.116 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLinesCutAtNodes (gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial operator: Line Cut At Nodes.

Parameters

geom1	pointer to input Geometry object [Linestring or MultiLinestring].
geom2	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.

5.5.2.117 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLineSubstring (gaiaGeomCollPtr *ln_geom*, double *start_fraction*, double *end_fraction*)

Spatial operator: Line Substring.

Parameters

ln_geom	the input Geometry object [expected to be of lineal type]
start_fraction	substring start, expressed as total length fraction [in the range 0.0 / 1.0]
end_fraction	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.

5.5.2.118 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLineSubstring_r (const void * *p_cache*, gaiaGeomCollPtr *ln_geom*, double *start_fraction*, double *end_fraction*)

Spatial operator: Line Substring.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
ln_geom	the input Geometry object [expected to be of lineal type]
start_fraction	substring start, expressed as total length fraction [in the range 0.0 / 1.0]
end_fraction	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.

5.5.2.119 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaMakeValid (gaiaGeomCollPtr *geom*)

Utility function: MakeValid.

Parameters

geom	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 loosing 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

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by gaiaMakeValid()

Remarks

LWGEOM support required.

5.5.2.120 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaMakeValidDiscarded (gaiaGeomCollPtr *geom*)

Utility function: MakeValidDiscarded.

Parameters

geom	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

LWGEOM support required.

5.5.2.121 GAIAGEO_DECLARE int gaiaMaxDistance (gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*, double * *dist*)

Calculates the maximum 2D distance intercurring between two Geometry objects.

Parameters

geom1	the first Geometry object
geom2	the second Geometry object
dist	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

LWGEOM support required.

5.5.2.122 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaNodeLines (gaiaGeomCollPtr *input*)

Utility function: re-noding lines.

Parameters

input	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 gaiaNode()

Remarks

LWGEOM support required.

5.5.2.123 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaOffsetCurve (gaiaGeomCollPtr *geom*, double *radius*, int *points*, int *left_right*)

Spatial operator: Offset Curve.

Parameters

geom	the input Geometry object
radius	the buffer's radius
points	number of points (aka vertices) to be used in order to approximate a circular arc.
left_right	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.

5.5.2.124 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaOffsetCurve_r (const void * *p_cache*, gaiaGeomCollPtr *geom*, double *radius*, int *points*, int *left_right*)

Spatial operator: Offset Curve.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	the input Geometry object
radius	the buffer's radius
points	number of points (aka vertices) to be used in order to approximate a circular arc.
left_right	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.

5.5.2.125 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaPolygonize (gaiaGeomCollPtr *geom*, int *force_multi*)

Attempts to rearrange a generic Geometry object into a Polygon or MultiPolygon.

Parameters

geom	the input Geometry object
force_multi	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.

5.5.2.126 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

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	the input Geometry object
force_multi	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.

5.5.2.127 GAIAGEO_DECLARE int gaiaProjectedPoint (double *x1*, double *y1*, double *a*, double *b*, double *distance*, double *azimuth*, double * *x2*, double * *y2*)

Utility function: ProjectedPoint.

Parameters

x1	the X coordinate of the Start Point.
y1	the Y coordinate of the Start Point.
a	major axis of the reference spheroid.
b	minor axis of the reference spheroid.
distance	a distance expressed in Meters
azimuth	(aka bearing aka heading) expressed in radians; on the clock: 12=0; 3=PI/2; 6=PI; 9=3PI/2.
x2	on completion this variable will contain the the X coordinate of the Projected Point.
y2	on completion this variable will contain the the Y coordinate of the Projected Point.

Returns

0 on failure: any other value on success

Remarks

LWGEOM support required.

5.5.2.128 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.

5.5.2.129 GAIAGEO_DECLARE void gaiaResetGeosMsg_r (const void * *p_cache*)

Resets the GEOS error and warning messages to an empty state.

Parameters

p_cache	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.

5.5.2.130 GAIAGEO_DECLARE void gaiaResetLwGeomMsg (void)

Resets the LWGEOM error and warning messages to an empty state.

See also

gaiaGetLwGeomErrorMsg, gaiaGetLwGeomWarningMsg, gaiaSetLwGeomErrorMsg, gaiaSetLwGeom↩WarningMsg

Note

not reentrant and thread unsafe.

Remarks

LWGEOM support required.

5.5.2.131 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSegmentize (gaiaGeomCollPtr *geom*, double *dist*)

Utility function: Segmentize.

Parameters

geom	the input Geometry object.
dist	the meximum 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

LWGEOM support required.

5.5.2.132 GAIAGEO_DECLARE void gaiaSetGeosAuxErrorMsg (const char * *msg*)

Set the current GEOS (auxiliary) error message.

Parameters

msg	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.

5.5.2.133 GAIAGEO_DECLARE void gaiaSetGeosAuxErrorMsg_r (const void * *p_cache,* const char * *msg*)

Set the current GEOS (auxiliary) error message.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
msg	the error message to be set.

See also

gaiaSetAuxErrorMsg, gaiaResetGeosMsg, gaiaGetGeosErrorMsg, gaiaGetGeosWarningMsg, gaiaGetGeosAuxErrorMsg, gaiaSetGeosWarningMsg, gaiaSetGeosErrorMsg

Note

reentrant and thread-safe.

Remarks

GEOS support required.

5.5.2.134 GAIAGEO_DECLARE void gaiaSetGeosErrorMsg (const char * *msg*)

Set the current GEOS error message.

Parameters

msg	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.

5.5.2.135 GAIAGEO_DECLARE void gaiaSetGeosErrorMsg_r (const void * *p_cache*, const char * *msg*)

Set the current GEOS error message.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
msg	the error message to be set.

See also

gaiaSetGeosErrorMsg, gaiaResetGeosMsg, gaiaGetGeosErrorMsg, gaiaGetGeosWarningMsg, gaiaGetGeosAuxErrorMsg, gaiaSetGeosWarningMsg, gaiaSetGeosAuxErrorMsg

Note

reentrant and thread-safe.

Remarks

GEOS support required.

5.5.2.136 GAIAGEO_DECLARE void gaiaSetGeosWarningMsg (const char * *msg*)

Set the current GEOS warning message.

Parameters

msg	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.

5.5.2.137 GAIAGEO_DECLARE void gaiaSetGeosWarningMsg_r (const void * *p_cache*, const char * *msg*)

Set the current GEOS warning message.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
msg	the warning message to be set.

See also

gaiaSetGeosWarningMsg, gaiaResetGeosMsg, gaiaGetGeosErrorMsg, gaiaGetGeosWarningMsg, gaiaGetGeosAuxErrorMsg, gaiaSetGeosErrorMsg, gaiaSetGeosAuxErrorMsg

Note

reentrant and thread-safe.

Remarks

GEOS support required.

5.5.2.138 GAIAGEO_DECLARE void gaiaSetLwGeomErrorMsg (const char * *msg*)

Set the current LWGEOM error message.

Parameters

msg	the error message to be set.

See also

gaiaResetLwGeomMsg, gaiaGetLwGeomErrorMsg, gaiaGetLwGeomWarningMsg, gaiaSetLwGeom↩WarningMsg

Note

not reentrant and thread unsafe.

Remarks

LWGEOM support required.

5.5.2.139 GAIAGEO_DECLARE void gaiaSetLwGeomWarningMsg (const char * *msg*)

Set the current LWGEOM warning message.

Parameters

msg	the warning message to be set.

See also

gaiaResetLwGeomMsg, gaiaGetLwGeomErrorMsg, gaiaGetLwGeomWarningMsg, gaiaSetLwGeomError↩Msg

Note

not reentrant and thread unsafe.

Remarks

LWGEOM support required.

5.5.2.140 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSharedPaths (gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial operator: Shared Paths.

Parameters

geom1	pointer to first Geometry object
geom2	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.

5.5.2.141 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSharedPaths_r (const void * *p_cache*, gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial operator: Shared Paths.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	pointer to first Geometry object
geom2	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.

5.5.2.142 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaShortestLine (gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial operator: Shortest Line.

Parameters

geom1	pointer to the first Geometry object.
geom2	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 intercurrinng 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.

5.5.2.143 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaShortestLine_r (const void * *p_cache*, gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Spatial operator: Shortest Line.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	pointer to the first Geometry object.
geom2	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 intercurrinng 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.

5.5.2.144 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSingleSidedBuffer (gaiaGeomCollPtr *geom*, double *radius*, int *points*, int *left_right*)

Spatial operator: Single Sided Buffer.

Parameters

geom	the input Geometry object
radius	the buffer's radius
points	number of points (aka vertices) to be used in order to approximate a circular arc.
left_right	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.

5.5.2.145 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSingleSidedBuffer_r (const void * *p_cache*, gaiaGeomCollPtr *geom*, double *radius*, int *points*, int *left_right*)

Spatial operator: Single Sided Buffer.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	the input Geometry object
radius	the buffer's radius
points	number of points (aka vertices) to be used in order to approximate a circular arc.
left_right	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.

5.5.2.146 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSnap (gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*, double *tolerance*)

Spatial operator: Snap.

Parameters

geom1	pointer to the first Geometry object.
geom2	pointer to the second Geometry object.
tolerance	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.

5.5.2.147 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSnap_r (const void * *p_cache*, gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*, double *tolerance*)

Spatial operator: Snap.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	pointer to the first Geometry object.
geom2	pointer to the second Geometry object.
tolerance	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.

5.5.2.148 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.

Parameters

geom	the input Geometry object.
origin_x	the X ccordinate identifying the Grid Origin.
origin_y	the Y coordinate identifiying the Grid Origin.
origin_z	the Z ccordinate identifying the Grid Origin.
origin_m	the M coordinate identifiying the Grid Origin.
size_x	Grid cell size (X axis).
size_y	Grid cell size (Y axis).
size_z	Grid cell size (Z axis).
size_m	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()

5.5.2.149 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSplit (gaiaGeomCollPtr *input,* gaiaGeomCollPtr *blade*)

Utility function: Split.

Parameters

input	the input Geometry object.
blade	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

LWGEOM support required.

5.5.2.150 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSplitLeft (gaiaGeomCollPtr *input,* gaiaGeomCollPtr *blade*)

Utility function: SplitLeft.

Parameters

input	the input Geometry object.
blade	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

LWGEOM support required.

5.5.2.151 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSplitRight (gaiaGeomCollPtr *input*, gaiaGeomCollPtr *blade*)

Utility function: SplitRight.

Parameters

input	the input Geometry object.
blade	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

LWGEOM support required.

5.5.2.152 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSquareGrid (gaiaGeomCollPtr *geom*, double *origin_x*, double *origin_y*, double *size*, int *only_edges*)

Utility function: SquareGrid.

Parameters

geom	the Geometry to be covered by the Grid.
origin_x	the X ccordinate identifying the Grid Origin.
origin_y	the Y coordinate identifiying the Grid Origin.
size	the Grid cell-side size.
only_edges	if non-zero will return a MULTILINESTRING, otherwise it 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.

5.5.2.153 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSquareGrid_r (const void * *p_cache*, gaiaGeomCollPtr *geom*, double *origin_x*, double *origin_y*, double *size*, int *only_edges*)

Utility function: SquareGrid.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	the Geometry to be covered by the Grid.
origin_x	the X ccordinate identifying the Grid Origin.
origin_y	the Y coordinate identifiying the Grid Origin.
size	the Grid cell-side size.
only_edges	if non-zero will return a MULTILINESTRING, otherwise it 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.

5.5.2.154 GAIAGEO_DECLARE void* gaiaToGeos (const gaiaGeomCollPtr *gaia*)

Converts a Geometry object into a GEOS Geometry.

Parameters

gaia	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.

5.5.2.155 GAIAGEO_DECLARE void* gaiaToGeos_r (const void * *p_cache,* const gaiaGeomCollPtr *gaia*)

Converts a Geometry object into a GEOS Geometry.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
gaia	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.

5.5.2.156 GAIAGEO_DECLARE void* gaiaToGeosSelective (const gaiaGeomCollPtr *gaia,* int *mode*)

Converts a Geometry object into a GEOS Geometry.

Parameters

gaia	pointer to Geometry object
mode	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.

5.5.2.157 GAIAGEO_DECLARE void* gaiaToGeosSelective_r (const void * *p_cache*, const gaiaGeomCollPtr *gaia*, int *mode*)

Converts a Geometry object into a GEOS Geometry.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
gaia	pointer to Geometry object
mode	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.

5.5.2.158 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaTriangularGrid (gaiaGeomCollPtr *geom*, double *origin_x*, double *origin_y*, double *size*, int *only_edges*)

Utility function: TriangularGrid.

Parameters

geom	the Geometry to be covered by the Grid.
origin_x	the X ccordinate identifying the Grid Origin.
origin_y	the Y coordinate identifiying the Grid Origin.
size	the Grid cell-side size.
only_edges	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.
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.

5.5.2.159 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaTriangularGrid_r (const void * *p_cache*, gaiaGeomCollPtr *geom*, double *origin_x*, double *origin_y*, double *size*, int *only_edges*)

Utility function: TriangularGrid.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	the Geometry to be covered by the Grid.
origin_x	the X ccordinate identifying the Grid Origin.
origin_y	the Y coordinate identifiying the Grid Origin.
size	the Grid cell-side size.
only_edges	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.
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.

5.5.2.160 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaUnaryUnion (gaiaGeomCollPtr *geom*)

Spatial operator: Unary Union.

Parameters

geom	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.

5.5.2.161 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaUnaryUnion_r (const void * *p_cache,* gaiaGeomCollPtr *geom*)

Spatial operator: Unary Union.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	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.

5.5.2.162 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaUnionCascaded (gaiaGeomCollPtr *geom*)

Spatial operator: Union Cascaded.

Parameters

geom	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.

5.5.2.163 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaUnionCascaded_r (const void * *p_cache,* gaiaGeomCollPtr *geom*)

Spatial operator: Union Cascaded.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	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.

5.5.2.164 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaVoronojDiagram (gaiaGeomCollPtr *geom,* double *extra_frame_size,* double *tolerance,* int *only_edges*)

Voronoj Diagram.

Parameters

geom	pointer to input Geometry object.
extra_frame_size	percent factor expanding the BBOX of input Geometry
tolerance	optional snapping tolerance.
only_edges	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

gaiaVoronojDiagram_r, gaiaFreeGeomColl, gaiaDelaunayTriangulation

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by gaiaVoronojDiagram()
not reentrant and thread unsafe.

Remarks

GEOS-TRUNK support required.

5.5.2.165 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaVoronojDiagram_r (const void * *p_cache*, gaiaGeomCollPtr *geom*, double *extra_frame_size*, double *tolerance*, int *only_edges*)

Voronoj Diagram.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	pointer to input Geometry object.
extra_frame_size	percent factor expanding the BBOX of input Geometry
tolerance	optional snapping tolerance.
only_edges	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

gaiaVoronojDiagram, gaiaFreeGeomColl, gaiaDelaunayTriangulation

Note

you are responsible to destroy (before or after) any allocated Geometry, this including any Geometry returned by gaiaVoronojDiagram_r()
reentrant and thread-safe.

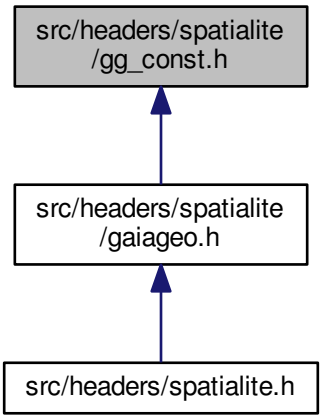
Remarks

GEOS-TRUNK support required.

5.6 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_MULTIPOINT 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_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_MULTIPOINTZ 1004

  BLOB-Geometry CLASS: MULTIPOINT 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_MULTIPOINTM 2004

  BLOB-Geometry CLASS: MULTIPOINT 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_MULTIPOINTZM 3004

  BLOB-Geometry CLASS: MULTIPOINT 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_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_MULTIPOINTZ -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_MULTIPOINT 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_MULTIPOINTZ 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_MULTIPOINTM 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_LHR_ORDER -2

  Clone Special Mode: apply Left Handle Rule to Polygon Rings.

- #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

5.6.1 Detailed Description

Geometry constants and macros.

5.6.2 Macro Definition Documentation

5.6.2.1 #define gaiaGetPoint(*xy, v, x, y*)

Value:

```
{*x = xy[(v) * 2]; \
                                    *y = xy[(v) * 2 + 1];}
```

macro extracting XY coordinates

Parameters

xy	pointer [const void ∗] to COORD mem-array
v	[int] point index [first point has index 0]
x	[double ∗] X coordinate
y	[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.

5.6.2.2 #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

xym	pointer [const void ∗] to COORD mem-array
v	[int] point index [first point has index 0]
x	[double ∗] X coordinate
y	[double ∗] Y coordinate
m	[double ∗] M measure

See also

gaiaLineGetPoint, gaiaRingGetPoint

Note

using this macro on behalf of COORDs not of [XYM] dims may cause serious problems

5.6.2.3 #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

xyz	pointer [const void *] to COORD mem-array
v	[int] point index [first point has index 0]
x	[double *] X coordinate
y	[double *] Y coordinate
z	[double *] Z coordinate

See also

gaiaLineGetPoint, gaiaRingGetPoint

Note

using this macro on behalf of COORDs not of [XYZ] dims may cause serious problems

5.6.2.4 #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

xyzm	pointer [const void *] to COORD mem-array
v	[int] point index [first point has index 0]
x	[double *] X coordinate
y	[double *] Y coordinate
z	[double *] Z coordinate
m	[double *] M measure

See also

gaiaLineGetPoint, gaiaRingGetPoint

Note

using this macro on behalf of COORDs not of [XYZM] dims may cause serious problems

5.6.2.5 #define gaiaSetPoint(*xy, v, x, y*)

Value:

```
{xy[(v) * 2] = x; \
                                xy[(v) * 2 + 1] = y;}
```

macro setting XY coordinates

Parameters

xy	pointer [const void ∗] to COORD mem-array
v	[int] point index [first point has index 0]
x	[double] X coordinate
y	[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.

5.6.2.6 #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

xym	pointer [const void ∗] to COORD mem-array
v	[int] point index [first point has index 0]
x	[double] X coordinate
y	[double] Y coordinate
m	[double] M measure

See also

gaiaLineSetPoint, gaiaRingSetPoint

Note

using this macro on behalf on COORDs not of [XYM] dims may cause serious problems

5.6.2.7 #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

xyz	pointer [const void ∗] to COORD mem-array
v	[int] point index [first point has index 0]
x	[double] X coordinate
y	[double] Y coordinate
z	[double] Z coordinate

See also

gaiaLineSetPoint, gaiaRingSetPoint

Note

using this macro on behalf on COORDs not of [XYZ] dims may cause serious problems

5.6.2.8 #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

xyzm	pointer [const void ∗] to COORD mem-array
v	[int] point index [first point has index 0]
x	[double] X coordinate
y	[double] Y coordinate
z	[double] Z coordinate
m	[double] M measure

See also

gaiaLineSetPoint, gaiaRingSetPoint

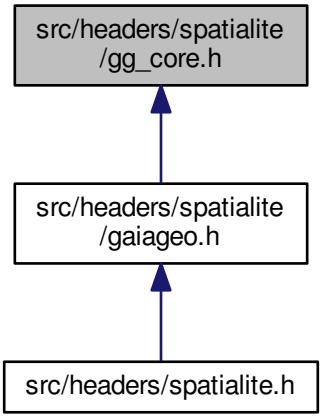
Note

using this macro on behalf on COORDs not of [XYZM] dims may cause serious problems

5.7 src/headers/spatialite/gg_core.h File Reference

Geometry handling functions: core.

This graph shows which files directly or indirectly include this file:



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 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 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 int gaiaLineGetPoint (gaiaLinestringPtr ln, int v, double ∗x, double ∗y, double ∗z, double ∗m)

  Gets coodinates 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 coodinates for a Ring's Point.
- GAIAGEO_DECLARE int gaiaDimension (gaiaGeomCollPtr geom)

  Determines OGC dimensions for a Geometry object.
- GAIAGEO_DECLARE int gaiaGeometryType (gaiaGeomCollPtr geom)

  Determines the corresponding Type for a Geometry object.
- GAIAGEO_DECLARE int gaiaGeometryAliasType (gaiaGeomCollPtr geom)

  Determines the corresponding Type for a Geometry object.
- GAIAGEO_DECLARE int gaiaIsEmpty (gaiaGeomCollPtr geom)

  Checks for empty Geometry object.
- GAIAGEO_DECLARE int gaiaIsToxic (gaiaGeomCollPtr geom)

  Checks for toxic Geometry object.
- GAIAGEO_DECLARE int gaiaIsToxic_r (const void *p_cache, gaiaGeomCollPtr geom)

  Checks for toxic Geometry object.
- GAIAGEO_DECLARE int gaiaIsNotClosedRing (gaiaRingPtr ring)

  Checks for not-closed Rings.
- GAIAGEO_DECLARE int gaiaIsNotClosedRing_r (const void *p_data, gaiaRingPtr ring)

  Checks for not-closed Rings.
- GAIAGEO_DECLARE int gaiaIsNotClosedGeomColl (gaiaGeomCollPtr geom)

  Checks for not-closed Rings in a Geometry object.
- GAIAGEO_DECLARE int gaiaIsNotClosedGeomColl_r (const void *p_data, gaiaGeomCollPtr geom)

  Checks for not-closed Rings in a Geometry object.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSanitize (gaiaGeomCollPtr org)

  Attempts to sanitize a possibly malformed Geometry object.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLinearize (gaiaGeomCollPtr geom, int force_multi)

  Attempts to resolve a (Multi)Linestring from a Geometry object.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaDissolveSegments (gaiaGeomCollPtr geom)

  Attempts to resolve a collection of Segments from a Geometry object.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaDissolvePoints (gaiaGeomCollPtr geom)

  Attempts to resolve a collection of Points from a Geometry object.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaExtractPointsFromGeomColl (gaiaGeomCollPtr geom)

  Extracts any Point from a Geometry object.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaExtractLinestringsFromGeomColl (gaiaGeomCollPtr geom)

  Extracts any Linestring from a Geometry object.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaExtractPolygonsFromGeomColl (gaiaGeomCollPtr geom)

  Extracts any Polygon from a Geometry object.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaMergeGeometries (gaiaGeomCollPtr geom1, gaiaGeomCollPtr geom2)

  Merges two Geometry objects into a single one.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaMergeGeometries_r (const void *p_cache, gaiaGeomCollPtr geom1, gaiaGeomCollPtr geom2)

  Merges two Geometry objects into a single one.
- 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 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 intesection 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 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 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 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.

5.7.1 Detailed Description

Geometry handling functions: core.

5.7.2 Function Documentation

5.7.2.1 GAIAGEO_DECLARE gaiaRingPtr gaiaAddInteriorRing (gaiaPolygonPtr *p*, int *pos*, int *vert*)

Creates a new Interior Ring object into a Polygon object.

Parameters

p	pointer to the Polygon object.
pos	relative position index [first Interior Ring has index 0].
vert	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.

5.7.2.2 GAIAGEO_DECLARE gaiaLinestringPtr gaiaAddLinestringToGeomColl (gaiaGeomCollPtr *p*, int *vert*)

Creates a new Linestring object into a Geometry object.

Parameters

p	pointer to the Geometry object.
vert	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.

5.7.2.3 GAIAGEO_DECLARE void gaiaAddPointToGeomColl (gaiaGeomCollPtr *p*, double *x*, double *y*)

Creates a new 2D Point [XY] object into a Geometry object.

Parameters

p	pointer to the Geometry object
x	X coordinate of the Point to be created
y	X 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.

5.7.2.4 GAIAGEO_DECLARE void gaiaAddPointToGeomCollXYM (gaiaGeomCollPtr *p*, double *x*, double *y*, double *m*)

Creates a new 2D Point [XYM] object into a Geometry object.

Parameters

p	pointer to the Geometry object
x	X coordinate of the Point to be created
y	X coordinate of the Point to be created
m	M measure of the Point to be created

Note

ownership of the newly created POINT object belongs to the Geometry object.

5.7.2.5 GAIAGEO_DECLARE void gaiaAddPointToGeomCollXYZ (gaiaGeomCollPtr *p*, double *x*, double *y*, double *z*)

Creates a new 3D Point [XYZ] object into a Geometry object.

Parameters

p	pointer to the Geometry object
x	X coordinate of the Point to be created
y	X coordinate of the Point to be created
z	Z coordinate of the Point to be created

Note

ownership of the newly created POINT object belongs to the Geometry object.

5.7.2.6 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.

Parameters

p	pointer to the Geometry object
x	X coordinate of the Point to be created
y	X coordinate of the Point to be created
z	Z coordinate of the Point to be created
m	M measure of the Point to be created

Note

ownership of the newly created POINT object belongs to the Geometry object.

5.7.2.7 GAIAGEO_DECLARE gaiaPolygonPtr gaiaAddPolygonToGeomColl (gaiaGeomCollPtr *p*, int *vert*, int *interiors*)

Creates a new Polygon object into a Geometry object.

Parameters

p	pointer to the Geometry object.
vert	number of points [aka vertices] into the Polygon's Exterior Ring.
interiors	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.

5.7.2.8 GAIAGEO_DECLARE void gaiaAddRingToPolyg (gaiaPolygonPtr *polyg*, gaiaRingPtr *ring*)

Inserts an already existing Ring object into a Polygon object.

Parameters

polyg	pointer to the Polygon object
ring	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.

5.7.2.9 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.

5.7.2.10 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.

5.7.2.11 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.

5.7.2.12 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.

5.7.2.13 GAIAGEO_DECLARE gaiaLinestringPtr gaiaAllocLinestring (int *vert*)

Allocates a 2D LINESTRING [XY].

Parameters

vert	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.

5.7.2.14 GAIAGEO_DECLARE gaiaLinestringPtr gaiaAllocLinestringXYM (int *vert*)

Allocates a 2D LINESTRING [XYM].

Parameters

vert	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.

5.7.2.15 GAIAGEO_DECLARE gaiaLinestringPtr gaiaAllocLinestringXYZ (int *vert*)

Allocates a 3D LINESTRING [XYZ].

Parameters

vert	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.

5.7.2.16 GAIAGEO_DECLARE gaiaLinestringPtr gaiaAllocLinestringXYZM (int *vert*)

Allocates a 3D LINESTRING [XYZM].

Parameters

vert	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.

5.7.2.17 GAIAGEO_DECLARE gaiaPointPtr gaiaAllocPoint (double *x*, double *y*)

Allocates a 2D POINT [XY].

Parameters

x	the X coordinate.
y	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.

5.7.2.18 GAIAGEO_DECLARE gaiaPointPtr gaiaAllocPointXYM (double *x*, double *y*, double *m*)

Allocates a 2D POINT [XYM].

Parameters

x	the X coordinate.
y	the Y coordinate.
m	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.

5.7.2.19 GAIAGEO_DECLARE gaiaPointPtr gaiaAllocPointXYZ (double *x*, double *y*, double *z*)

Allocates a 3D POINT [XYZ].

Parameters

x	the X coordinate.
y	the Y coordinate.
z	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.

5.7.2.20 GAIAGEO_DECLARE gaiaPointPtr gaiaAllocPointXYZM (double *x*, double *y*, double *z*, double *m*)

Allocates a 3D POINT [XYZM].

Parameters

x	the X coordinate.
y	the Y coordinate.
z	the Z coordinate.
m	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.

5.7.2.21 GAIAGEO_DECLARE gaiaPolygonPtr gaiaAllocPolygon (int *vert*, int *holes*)

Allocates a 2D POLYGON [XY].

Parameters

vert	number of points [aka vertices] into the Exterior Ring.
holes	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.

5.7.2.22 GAIAGEO_DECLARE gaiaPolygonPtr gaiaAllocPolygonXYM (int *vert*, int *holes*)

Allocates a 2D POLYGON [XYM].

Parameters

vert	number of points [aka vertices] into the Exterior Ring.
holes	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.

5.7.2.23 GAIAGEO_DECLARE gaiaPolygonPtr gaiaAllocPolygonXYZ (int *vert*, int *holes*)

Allocates a 3D POLYGON [XYZ].

Parameters

vert	number of points [aka vertices] into the Exterior Ring.
holes	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.

5.7.2.24 GAIAGEO_DECLARE gaiaPolygonPtr gaiaAllocPolygonXYZM (int *vert,* int *holes*)

Allocates a 3D POLYGON [XYZM].

Parameters

vert	number of points [aka vertices] into the Exterior Ring.
holes	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.

5.7.2.25 GAIAGEO_DECLARE gaiaRingPtr gaiaAllocRing (int *vert*)

Allocates a 2D RING [XY].

Parameters

vert	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.

5.7.2.26 GAIAGEO_DECLARE gaiaRingPtr gaiaAllocRingXYM (int *vert*)

Allocates 2D RING [XYM].

Parameters

vert	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.

5.7.2.27 GAIAGEO_DECLARE gaiaRingPtr gaiaAllocRingXYZ (int *vert*)

Allocates a 3D RING [XYZ].

Parameters

vert	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.

5.7.2.28 GAIAGEO_DECLARE gaiaRingPtr gaiaAllocRingXYZM (int *vert*)

Allocates a 3D RING [XYZM].

Parameters

vert	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.

5.7.2.29 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaCastGeomCollToXY (gaiaGeomCollPtr *geom*)

Duplicates a Geometry object [casting dimensions to 2D XY].

Parameters

geom	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.

5.7.2.30 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaCastGeomCollToXYM (gaiaGeomCollPtr *geom*)

Duplicates a Geometry object [casting dimensions to 2D XYM].

Parameters

geom	pointer to Geometry object [origin].

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

gaiaCloneGeomColl, gaiaCastGeomCollToXY, gaiaCastGeomCollToXYZ, gaiaCastGeomCollToXYZM

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.

5.7.2.31 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaCastGeomCollToXYZ (gaiaGeomCollPtr *geom*)

Duplicates a Geometry object [casting dimensions to 3D XYZ].

Parameters

geom	pointer to Geometry object [origin].

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

gaiaCloneGeomColl, gaiaCastGeomCollToXY, gaiaCastGeomCollToXYM, 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 [XYZ] dimensions.

5.7.2.32 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaCastGeomCollToXYZM (gaiaGeomCollPtr *geom*)

Duplicates a Geometry object [casting dimensions to 3D XYZM].

Parameters

geom	pointer to Geometry object [origin].

Returns

the pointer to newly created Geometry object: NULL on failure.

See also

gaiaCloneGeomColl, gaiaCastGeomCollToXY, gaiaCastGeomCollToXYZ, gaiaCastGeomCollToXYM

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.

5.7.2.33 GAIAGEO_DECLARE void gaiaClockwise (gaiaRingPtr *p*)

Determines the direction for a Ring object.

Parameters

p	pointer to Ring object

Returns

0 if the ring has counter-clockwise direction; any other different value for clockwise direction.

5.7.2.34 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaCloneGeomColl (gaiaGeomCollPtr *geom*)

Duplicates a Geometry object.

Parameters

geom	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.

5.7.2.35 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaCloneGeomCollLinestrings (gaiaGeomCollPtr *geom*)

Duplicates a Geometry object [Linestrings only].

Parameters

geom	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.

5.7.2.36 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaCloneGeomCollPoints (gaiaGeomCollPtr *geom*)

Duplicates a Geometry object [Points only].

Parameters

geom	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.

5.7.2.37 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaCloneGeomCollPolygons (gaiaGeomCollPtr *geom*)

Duplicates a Geometry object [Polygons only].

Parameters

geom	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.

5.7.2.38 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaCloneGeomCollSpecial (gaiaGeomCollPtr *geom*, int *mode*)

Duplicates a Geometry object (special)

Parameters

geom	pointer to Geometry object [origin].
mode	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.

5.7.2.39 GAIAGEO_DECLARE gaiaLinestringPtr gaiaCloneLinestring (gaiaLinestringPtr *line*)

Duplicates a Linestring object.

Parameters

line	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.

5.7.2.40 GAIAGEO_DECLARE gaiaLinestringPtr gaiaCloneLinestringSpecial (gaiaLinestringPtr *line*, int *mode*)

Duplicates a Linestring object (special)

Parameters

line	pointer to Linestring object [origin].
mode	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.

5.7.2.41 GAIAGEO_DECLARE gaiaPolygonPtr gaiaClonePolygon (gaiaPolygonPtr *polyg*)

Duplicates a Polygon object.

Parameters

polyg	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.

5.7.2.42 GAIAGEO_DECLARE gaiaPolygonPtr gaiaClonePolygonSpecial (gaiaPolygonPtr *polyg*, int *mode*)

Duplicates a Polygon object (special)

Parameters

polyg	pointer to Polygon object [origin].
mode	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_LHR_ORDER is specified instead, any Exterior Ring will have clockwise orientation, and any Interior Ring will have counter-clockwise orientation. In any other case this function will simply default to gaiaClone↩ Polygon.

5.7.2.43 GAIAGEO_DECLARE gaiaRingPtr gaiaCloneRing (gaiaRingPtr *ring*)

Duplicates a Ring object.

Parameters

ring	pointer to Ring object [origin].

Returns

the pointer to newly created Ring object: NULL on failure.

See also

gaiaCloneLinestring, gaiaClonePolygon, gaiaCloneGeomColl, gaiaCloneGeomCollPoints, gaiaCloneGeom↩ CollLinestrings, gaiaCloneGeomCollPolygons, gaiaCloneRingSpecial

Note

the newly created object is an exact copy of the original one.

5.7.2.44 GAIAGEO_DECLARE gaiaRingPtr gaiaCloneRingSpecial (gaiaRingPtr *ring,* int *mode*)

Duplicates a Ring object (special)

Parameters

ring	pointer to Ring object [origin].
mode	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.

5.7.2.45 GAIAGEO_DECLARE int gaiaConvertLength (double *value,* int *unit_from,* int *unit_to,* double * *cvt*)

Convert a Length from a Measure Unit to another.

Parameters

value	the length measure to be converted.
unit_from	original Measure Unit.
unit_to	converted Measure Unit.
cvt	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

5.7.2.46 GAIAGEO_DECLARE void gaiaCopyLinestringCoords (gaiaLinestringPtr *dst*, gaiaLinestringPtr *src*)

Copies coordinates between two LINESTRING objects.

Parameters

dst	destination LINESTRING [output]
src	origin LINESTRING [input]

See also

gaiaCopyLinestringCoordsReverse

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.

5.7.2.47 GAIAGEO_DECLARE void gaiaCopyLinestringCoordsReverse (gaiaLinestringPtr *dst*, gaiaLinestringPtr *src*)

Copies coordinates between two LINESTRING objects in reverse order.

Parameters

dst	destination LINESTRING [output]
src	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.

5.7.2.48 GAIAGEO_DECLARE void gaiaCopyRingCoords (gaiaRingPtr *dst*, gaiaRingPtr *src*)

Copies coordinates between two RING objects.

Parameters

dst	destination RING [output]
src	origin RING [input]

See also

gaiaCopyRingCoordsReverse

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.

5.7.2.49 GAIAGEO_DECLARE void gaiaCopyRingCoordsReverse (gaiaRingPtr *dst*, gaiaRingPtr *src*)

Copies coordinates between two RING objects in reverse order.

Parameters

dst	destination RING [output]
src	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.

5.7.2.50 GAIAGEO_DECLARE gaiaPolygonPtr gaiaCreatePolygon (gaiaRingPtr *ring*)

Allocates a POLYGON.

Parameters

ring	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.

5.7.2.51 GAIAGEO_DECLARE int gaiaDimension (gaiaGeomCollPtr *geom*)

Determines OGC dimensions for a Geometry object.

Parameters

geom	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.

5.7.2.52 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaDissolvePoints (gaiaGeomCollPtr *geom*)

Attempts to resolve a collection of Points from a Geometry object.

Parameters

geom	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.

5.7.2.53 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaDissolveSegments (gaiaGeomCollPtr *geom*)

Attempts to resolve a collection of Segments from a Geometry object.

Parameters

geom	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.

5.7.2.54 GAIAGEO_DECLARE int gaiaEllipseParams (const char * *name*, double * *a*, double * *b*, double * *rf*)

Retrieves Geodesic params for an Ellipsoid definition.

Parameters

name	text string identifying an Ellipsoid definition.
a	on completion this variable will contain the first geodesic param.
b	on completion this variable will contain the second geodesic param.
rf	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**, **andrae**, **aust_SA**, **GRS67**, **bessel**, **bess_nam**, **clrk66**, **clrk80**, **CPM**, **delmbr**, **engelis**, **evrst30**, **evrst48**, **evrst56**, **evrst69**, **evrstSS**, **fschr60**

5.7.2.55 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaExtractLinestringsFromGeomColl (gaiaGeomCollPtr *geom*)

Extracts any Linestring from a Geometry object.

Parameters

geom	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.

5.7.2.56 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaExtractPointsFromGeomColl (gaiaGeomCollPtr *geom*)

Extracts any Point from a Geometry object.

Parameters

geom	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.

5.7.2.57 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaExtractPolygonsFromGeomColl (gaiaGeomCollPtr *geom*)

Extracts any Polygon from a Geometry object.

Parameters

geom	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.

5.7.2.58 GAIAGEO_DECLARE void gaiaFree (void * *ptr*)

Safely frees any dynamic memory block allocated by the library itself.

Parameters

ptr	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.

5.7.2.59 GAIAGEO_DECLARE void gaiaFreeGeomColl (gaiaGeomCollPtr *geom*)

Destroys a Geometry object.

Parameters

geom	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.

5.7.2.60 GAIAGEO_DECLARE void gaiaFreeLinestring (gaiaLinestringPtr *ptr*)

Destroys a LINESTRING object.

Parameters

ptr	pointer to the LINESTRING object to be destroyed

See also

gaiaAllocLinestring, gaiaAllocLinestringXYZ, gaiaAllocLinestringXYM, gaiaAllocLinestringXYZM

Note

attempting to destroy any LINESTRING object whose ownnership has already been transferred to some other (higher order) object is a serious error, and will easily cause severe memory corruption.

5.7.2.61 GAIAGEO_DECLARE void gaiaFreePoint (gaiaPointPtr *ptr*)

Destroys a POINT object.

Parameters

ptr	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.

5.7.2.62 GAIAGEO_DECLARE void gaiaFreePolygon (gaiaPolygonPtr *polyg*)

Destroys a POLYGON object.

Parameters

polyg	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.

5.7.2.63 GAIAGEO_DECLARE void gaiaFreeRing (gaiaRingPtr *ptr*)

Destroys a RING object.

Parameters

ptr	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.

5.7.2.64 GAIAGEO_DECLARE double gaiaGeodesicDistance (double *a*, double *b*, double *rf*, double *lat1*, double *lon1*, double *lat2*, double *lon2*)

Calculates the Geodesic Distance between between two Points.

Parameters

a	first geodesic parameter.
b	second geodesic parameter.
rf	third geodesic parameter.
lat1	Latitude of first Point.
lon1	Longitude of first Point.
lat2	Latitude of second Point.
lon2	Longitude of second Point.

Returns

the calculated Geodesic Distance.

See also

gaiaEllipseParams, gaiaGreatCircleDistance, gaiaGreatCircleTotalLength, gaiaGeodesicTotalLength

Note

the returned distance is expressed in Kilometers.
the Geodesic method is much more accurate but slowest to be calculated.

5.7.2.65 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.

Parameters

a	first geodesic parameter.
b	second geodesic parameter.
rf	third geodesic parameter.

dims	dimensions: one of GAIA_XY, GAIA_XY_Z, GAIA_XY_M or GAIA_XY_ZM
coords	pointed to COORD mem-array
vert	number of Points (aka Vertices) within the COORD mem-array

Returns

the calculated Geodesic Total Length.

See also

gaiaEllipseParams, gaiaGreatCircleDistance, gaiaGeodesicDistance, gaiaGreatCircleTotalLength

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

5.7.2.66 GAIAGEO_DECLARE int gaiaGeometryAliasType (gaiaGeomCollPtr *geom*)

Determines the corresponding Type for a Geometry object.

Parameters

geom	pointer to Geometry object

Returns

the corresponding Geometry Type

See also

gaiaGeometryType

Note

Type is one of: GAIA_POINT, GAIA_LINESTRING, GAIA_POLYGON, GAIA_MULTIPOINT, GAIA_MULTIL↩
INESTRING, GAIA_MULTIPOLYGON, GAIA_GEOMETRYCOLLECTION
on failure GAIA_NONE will be returned.

Remarks

deprecated function (used in earlier SpatiaLite versions).

5.7.2.67 GAIAGEO_DECLARE int gaiaGeometryType (gaiaGeomCollPtr *geom*)

Determines the corresponding Type for a Geometry object.

Parameters

geom	pointer to Geometry object

Returns

the corresponding Geometry Type

Note

Type is one of: GAIA_POINT, GAIA_LINESTRING, GAIA_POLYGON, GAIA_MULTIPOINT, GAIA_MULTI↩
LINESTRING, GAIA_MULTIPOLYGON, GAIA_GEOMETRYCOLLECTION, GAIA_POINTZ, GAIA_LINEST↩
RINGZ, GAIA_POLYGONZ, GAIA_MULTIPOINTZ, GAIA_MULTILINESTRINGZ, GAIA_MULTIPOLYGON↩
Z, GAIA_GEOMETRYCOLLECTIONZ, GAIA_POINTM, GAIA_LINESTRINGM, GAIA_POLYGONM, GAIA↩
_MULTIPOINTM, GAIA_MULTILINESTRINGM, GAIA_MULTIPOLYGONM, GAIA_GEOMETRYCOLLECT↩
IONM, GAIA_POINTZM, GAIA_LINESTRINGZM, GAIA_POLYGONZM, GAIA_MULTIPOINTZM, GAIA_M↩
ULTILINESTRINGZM, GAIA_MULTIPOLYGONZM, GAIA_GEOMETRYCOLLECTIONZM
on failure GAIA_NONE will be returned.

Examples:

demo1.c, and demo2.c.

5.7.2.68 GAIAGEO_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

a	first geodesic parameter.
b	second geodesic parameter.
lat1	Latitude of first Point.
lon1	Longitude of first Point.
lat2	Latitude of second Point.
lon2	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.

5.7.2.69 GAIAGEO_DECLARE double gaiaGreatCircleTotalLength (double *a*, double *b*, int *dims*, double * *coords*, int *vert*)

Calculates the Great Circle Total Length for a Linestring / Ring.

Parameters

a	first geodesic parameter.
b	second geodesic parameter.
dims	dimensions: one of GAIA_XY, GAIA_XY_Z, GAIA_XY_M or GAIA_XY_ZM
coords	pointed to COORD mem-array
vert	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

5.7.2.70 GAIAGEO_DECLARE void gaiaInsertInteriorRing (gaiaPolygonPtr *p*, gaiaRingPtr *ring*)

Inserts an already existing Ring object into a Polygon object.

Parameters

p	pointer to the Polygon object
ring	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.

5.7.2.71 GAIAGEO_DECLARE void gaiaInsertLinestringInGeomColl (gaiaGeomCollPtr *p*, gaiaLinestringPtr *line*)

Inserts an already existing Linestring object into a Geometry object.

Parameters

p	pointer to the Geometry object.
line	pointer to the Linestring object.

Note

ownership of the Linestring object will be transferred to the Geometry object.

5.7.2.72 GAIAGEO_DECLARE gaiaPolygonPtr gaiaInsertPolygonInGeomColl (gaiaGeomCollPtr *p*, gaiaRingPtr *ring*)

Creates a new Polygon object into a Geometry object starting from an already existing Ring object.

Parameters

p	pointer to the Geometry object.
ring	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.

5.7.2.73 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 intesection Point between two Segments.

Parameters

x0	on completion this variable will contain the Intersection X coord
y0	on completion this variable will contain the Intersection Y coord
x1	start Point X of first Segment
y1	start Point Y of first Segment
x2	end Point X of first Segment
y2	end Point Y of first Segment
x3	start Point X of second Segment
y3	start Point Y of second Segment
x4	end Point X of second Segment
y4	end Point Y of second Segment

Returns

0 if the Segments doesn't intersect at all: any other value on success.

5.7.2.74 GAIAGEO_DECLARE int gaiaIsEmpty (gaiaGeomCollPtr *geom*)

Checks for empty Geometry object.

Parameters

geom	pointer to Geometry object

Returns

0 if the Geometry is 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.

5.7.2.75 GAIAGEO_DECLARE int gaiaIsNotClosedGeomColl (gaiaGeomCollPtr *geom*)

Checks for not-closed Rings in a Geometry object.

Parameters

geom	pointer to Geometry object

Returns

0 if the Geometry has no unclosed Rings: otherwise any other different value.

See also

gaiaIsNotClosedGeomColl_r, gaiaIsToxic, gaiaIsNotClosedRing

Note

This function allows to explicitly identify any Geometry containing at least one unclosed Ring.
not reentrant and thread unsafe.

5.7.2.76 GAIAGEO_DECLARE int gaiaIsNotClosedGeomColl_r (const void * *p_data*, gaiaGeomCollPtr *geom*)

Checks for not-closed Rings in a Geometry object.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	pointer to Geometry object

Returns

0 if the Geometry has no unclosed Rings: otherwise any other different value.

See also

gaiaIsNotClosedGeomColl, gaiaIsToxic, gaiaIsNotClosedRing

Note

This function allows to explicitly identify any Geometry containing at least one unclosed Ring.
reentrant and thread-safe.

5.7.2.77 GAIAGEO_DECLARE int gaiaIsNotClosedRing (gaiaRingPtr *ring*)

Checks for not-closed Rings.

Parameters

ring	pointer to Ring object

Returns

0 if the Ring in unclosed: otherwise any other different value.

See also

gaiaIsNotClosedRing_r, gaiaIsToxic, gaiaIsNotClosedGeomColl

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.

5.7.2.78 GAIAGEO_DECLARE int gaiaIsNotClosedRing_r (const void * *p_data*, gaiaRingPtr *ring*)

Checks for not-closed Rings.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
ring	pointer to Ring object

Returns

0 if the Ring in unclosed: otherwise any other different value.

See also

gaiaIsNotClosedRing, gaiaIsToxic, gaiaIsNotClosedGeomColl

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.

5.7.2.79 GAIAGEO_DECLARE int gaiaIsPointOnPolygonSurface (gaiaPolygonPtr *polyg*, double *x*, double *y*)

Checks if a Point lays on a Polygon surface.

Parameters

polyg	pointer to Polygon object
x	Point X coordinate
y	Point Y coordinate

Returns

0 if false: any other value if true

5.7.2.80 GAIAGEO_DECLARE int gaiaIsPointOnRingSurface (gaiaRingPtr *ring*, double *pt_x*, double *pt_y*)

Check if a Point lays on a Ring surface.

Parameters

ring	pointer to Ring object
pt_x	Point X coordinate
pt_y	Point Y coordinate

Returns

0 if false: any other value if true

5.7.2.81 GAIAGEO_DECLARE int gaiaIsToxic (gaiaGeomCollPtr *geom*)

Checks for toxic Geometry object.

Parameters

geom	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.

5.7.2.82 GAIAGEO_DECLARE int gaiaIsToxic_r (const void * *p_cache*, gaiaGeomCollPtr *geom*)

Checks for toxic Geometry object.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom	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.

5.7.2.83 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLinearize (gaiaGeomCollPtr *geom,* int *force_multi*)

Attempts to resolve a (Multi)Linestring from a Geometry object.

Parameters

geom	pointer to Geometry object.
force_multi	0 if the returned Geometry could represent a Linestring: any other value if casting to Multi↩ Linestring 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.

5.7.2.84 GAIAGEO_DECLARE int gaiaLineGetPoint (gaiaLinestringPtr *ln*, int *v*, double * *x*, double * *y*, double * *z*, double * *m*)

Gets coodinates from a Linestring's Point.

Parameters

ln	pointer to Linestring object.
v	relative position of Point: first Point has index 0
x	on completion this variable will contain the Point X coordinate.
y	on completion this variable will contain the Point Y coordinate.
z	on completion this variable will contain the Point Z coordinate.
m	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(), gaiaGetPoint↩ XYM() 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.

5.7.2.85 GAIAGEO_DECLARE int gaiaLineSetPoint (gaiaLinestringPtr *ln*, int *v*, double *x*, double *y*, double *z*, double *m*)

Sets coordinates for a Linestring's Point.

Parameters

ln	pointer to Linestring object.
v	relative position of Point: first Point has index 0
x	the Point's X coordinate.
y	the Point's Y coordinate.
z	the Point's Z coordinate.
m	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(), gaiaSetPoint↩XYM() 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.

5.7.2.86 GAIAGEO_DECLARE int gaiaLinestringEquals (gaiaLinestringPtr *line1*, gaiaLinestringPtr *line2*)

Checks if two Linestring objects are equivalent.

Parameters

line1	pointer to first Linestring object.
line2	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.

5.7.2.87 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaLocateBetweenMeasures (gaiaGeomCollPtr *geom*, double *m_start*, double *m_end*)

Return a GeometryCollection containing elements matching the specified range of measures.

Parameters

geom	pointer to Geometry object
m_start	range of measures: start value
m_end	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.

5.7.2.88 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

center_x	center point X coordinate.
center_y	center point Y coordinate.
radius	the circle's radius.
start	the start angle (in degrees).
start	the stop angle (in degrees).
step	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

5.7.2.89 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaMakeCircle (double *center_x*, double *center_y*, double *radius*, double *step*)

Creates a Circle (Linestring) Geometry.

Parameters

center_x	center point X coordinate.
center_y	center point Y coordinate.
radius	the circle's radius.

step	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

5.7.2.90 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaMakeEllipse (double *center_x*, double *center_y*, double *x_axis*, double *y_axis*, double *step*)

Creates an Ellipse (Linestring) Geometry.

Parameters

center_x	center point X coordinate.
center_y	center point Y coordinate.
x_axis	the ellipses's X axis.
y_axis	the ellipses's Y axis.
step	angular distance (in degrees) between points on the ellipse.

See also

gaiaMakeEllipticArc, gaiaMakeCircle, gaiaMakeArc

5.7.2.91 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

center_x	center point X coordinate.
center_y	center point Y coordinate.
x_axis	the ellipses's X axis.
y_axis	the ellipses's Y axis.
start	the start angle (in degrees).
start	the stop angle (in degrees).
step	angular distance (in degrees) between points on the ellipse.

See also

gaiaMakeCircle, gaiaMakeEllipse, gaiaMakeEllipticArc

5.7.2.92 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaMakePolygon (gaiaGeomCollPtr *exterior*, gaiaGeomCollPtr *interiors*)

Creates a Polygon from closed Linestrings.

Parameters

exterior	a closed Linestring assumed to represent the Exterior Ring.
interiors	one (or more than one) clsed Linestrings assumed to represent all Interior Rings (could be a Linstring 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.

5.7.2.93 GAIAGEO_DECLARE double gaiaMeasureArea (gaiaRingPtr *ring*)

Measures the geometric area for a Ring object.

Parameters

ring	pointer to Ring object

Returns

the calculated geometric area

See also

gaiaGeomCollArea

Remarks

internal method: doesn't require any GEOS support.

5.7.2.94 GAIAGEO_DECLARE double gaiaMeasureLength (int *dims*, double * *coords*, int *vert*)

Measures the geometric length for a Linestring or Ring.

Parameters

dims	dimensions: one of GAIA_XY, GAIA_XY_Z, GAIA_XY_M or GAIA_XY_ZM
coords	pointed to COORD mem-array
vert	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.

5.7.2.95 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaMergeGeometries (gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Merges two Geometry objects into a single one.

Parameters

geom1	pointer to first Geometry object.
geom2	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.

5.7.2.96 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaMergeGeometries_r (const void * *p_cache*, gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*)

Merges two Geometry objects into a single one.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
geom1	pointer to first Geometry object.
geom2	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.

5.7.2.97 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.

Parameters

x0	Point X coordinate
y0	Point Y coordinate
dims	dimensions: one of GAIA_XY, GAIA_XY_Z, GAIA_XY_M or GAIA_XY_ZM
coords	pointed to COORD mem-array
vert	number of Points (aka Vertices) within the COORD mem-array

Returns

the calculated minumum distance.

Note

dims, **coords** and **vert** are usually expected to correspond to **DimensionModel**, **Coords** and **Points** members from a gaiaLinestringStruct or gaiaRingStruct

5.7.2.98 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).

Parameters

geom	pointer to Geometry object.

See also

gaiaScaleCoords, gaiaRotateCoords, gaiaReflectCoords, gaiaSwapCoords, gaiaShiftCoords3D, gaiaShiftLongitude

5.7.2.99 GAIAGEO_DECLARE int gaiaPolygonEquals (gaiaPolygonPtr *polyg1,* gaiaPolygonPtr *polyg2*)

Checks if two Polygons objects are equivalent.

Parameters

polyg1	pointer to first Polygon object.
polyg2	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).

5.7.2.100 GAIAGEO_DECLARE void gaiaReflectCoords (gaiaGeomCollPtr *geom*, int *x_axis*, int *y_axis*)

Reflects any coordinate within a Geometry object.

Parameters

geom	pointer to Geometry object.
x_axis	if set to 0, no X axis reflection will be applied: otherwise the X axis will be reflected.
y_axis	if set to 0, no Y axis reflection will be applied: otherwise the Y axis will be reflected.

See also

gaiaShiftCoords, gaiaScaleCoords, gaiaRotateCoords, gaiaSwapCoords

5.7.2.101 GAIAGEO_DECLARE void gaiaRingCentroid (gaiaRingPtr *ring*, double * *rx*, double * *ry*)

Determines the Centroid for a Ring object.

Parameters

ring	pointer to Ring object.
rx	on completion this variable will contain the centroid X coordinate.
ry	on completion this variable will contain the centroid Y coordinate.

See also

gaiaGeomCollCentroid

Remarks

internal method: doesn't require any GEOS support.

5.7.2.102 GAIAGEO_DECLARE int gaiaRingGetPoint (gaiaRingPtr *rng*, int *v*, double * *x*, double * *y*, double * *z*, double * *m*)

Gets coordinates from a Ring's Point.

Parameters

rng	pointer to Ring object.
v	relative position of Point: first Point has index 0
x	on completion this variable will contain the Point X coordinate.
y	on completion this variable will contain the Point Y coordinate.
z	on completion this variable will contain the Point Z coordinate.
m	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(), gaiaGetPoint↩XYM() 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.

5.7.2.103 GAIAGEO_DECLARE int gaiaRingSetPoint (gaiaRingPtr *rng*, int *v*, double *x*, double *y*, double *z*, double *m*)

Sets coodinates for a Ring's Point.

Parameters

rng	pointer to Ring object.
v	relative position of Point: first Point has index 0
x	the Point's X coordinate.
y	the Point's Y coordinate.
z	the Point's Z coordinate.
m	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(), gaiaSetPoint↩XYM() 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.

5.7.2.104 GAIAGEO_DECLARE void gaiaRotateCoords (gaiaGeomCollPtr *geom*, double *angle*)

Rotates any coordinate within a Geometry object.

Parameters

geom	pointer to Geometry object.
angle	rotation angle [expressed in Degrees].

See also

gaiaShiftCoords, gaiaScaleCoords, gaiaReflectCoords, gaiaSwapCoords

5.7.2.105 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaSanitize (gaiaGeomCollPtr *org*)

Attempts to sanitize a possibly malformed Geometry object.

Parameters

org	pointer to Geometry object.

Returns

the pointer to newly created Geometry: NULL on failure.

See also

gaiaIsToxic

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.

5.7.2.106 GAIAGEO_DECLARE void gaiaScaleCoords (gaiaGeomCollPtr *geom*, double *scale_x*, double *scale_y*)

Scales any coordinate within a Geometry object.

Parameters

geom	pointer to Geometry object.
scale_x	X axis scale factor.
scale_y	Y axis scale factor.

See also

gaiaShiftCoords, gaiaRotateCoords, gaiaReflectCoords, gaiaSwapCoords

5.7.2.107 GAIAGEO_DECLARE void gaiaShiftCoords (gaiaGeomCollPtr *geom*, double *shift_x*, double *shift_y*)

Shifts any coordinate within a Geometry object.

Parameters

geom	pointer to Geometry object.
shift_x	X axis shift factor.
shift_y	Y axis shift factor.

See also

gaiaScaleCoords, gaiaRotateCoords, gaiaReflectCoords, gaiaSwapCoords, gaiaShiftCoords3D, gaiaShift↩Longitude

5.7.2.108 GAIAGEO_DECLARE void gaiaShiftCoords3D (gaiaGeomCollPtr *geom*, double *shift_x*, double *shift_y*, double *shift_z*)

Shifts any coordinate within a 3D Geometry object.

Parameters

geom	pointer to Geometry object.
shift_x	X axis shift factor.
shift_y	Y axis shift factor.
shift_z	Z axis shift factor.

See also

gaiaScaleCoords, gaiaRotateCoords, gaiaReflectCoords, gaiaSwapCoords, gaiaShiftCoords, gaiaShift↩Longitude, gaiaNormalizeLonLat

5.7.2.109 GAIAGEO_DECLARE void gaiaShiftLongitude (gaiaGeomCollPtr *geom*)

Shifts negative longitudes.

Parameters

geom	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.

5.7.2.110 GAIAGEO_DECLARE void gaiaSwapCoords (gaiaGeomCollPtr *geom*)

Swaps any coordinate within a Geometry object.

Parameters

geom	pointer to Geometry object.

See also

gaiaShiftCoords, gaiaScaleCoords, gaiaRotateCoords, gaiaReflectCoords

Note

the X and Y axes will be swapped.

5.8 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.

5.8.1 Detailed Description

Geometry handling functions: DXF files.

5.8.2 Typedef Documentation

5.8.2.1 typedef gaiaDxfArc* gaiaDxfArcPtr

Typedef for DXF Arc object.

See also

gaiaDxfArc

5.8.2.2 typedef gaiaDxfBlock* gaiaDxfBlockPtr

Typedef for DXF Block object.

See also

gaiaDxfBlock

5.8.2.3 typedef gaiaDxfBoundaryPath* gaiaDxfBoundaryPathPtr

Typedef for DXF Boundary Path object.

See also

gaiaDxfBoundaryPath

5.8.2.4 typedef gaiaDxfCircle* gaiaDxfCirclePtr

Typedef for DXF Circle object.

See also

gaiaDxfCircle

5.8.2.5 typedef gaiaDxfExtraAttr* gaiaDxfExtraAttrPtr

Typedef for DXF Extra Attribute object.

See also

gaiaDxfExtraAttr

5.8.2.6 typedef gaiaDxfHatch* gaiaDxfHatchPtr

Typedef for DXF Hatch object.

See also

gaiaDxfHatch

5.8.2.7 typedef gaiaDxfHatchSegm* gaiaDxfHatchSegmPtr

Typedef for DXF Hatch Segment object.

See also

gaiaDxfHatch

5.8.2.8 typedef gaiaDxfHole* gaiaDxfHolePtr

Typedef for DXF Point object.

See also

gaiaDxfHole

5.8.2.9 typedef gaiaDxfInsert* gaiaDxfInsertPtr

Typedef for DXF Insert object.

See also

gaiaDxfText

5.8.2.10 typedef gaiaDxfLayer* gaiaDxfLayerPtr

Typedef for DXF Layer object.

See also

gaiaDxfLayer

5.8.2.11 typedef gaiaDxfParser* gaiaDxfParserPtr

Typedef for DXF Layer object.

See also

gaiaDxfParser

5.8.2.12 typedef gaiaDxfPoint* gaiaDxfPointPtr

Typedef for DXF Point object.

See also

gaiaDxfPoint

5.8.2.13 typedef gaiaDxfPolyline* gaiaDxfPolylinePtr

Typedef for DXF Polyline object.

See also

gaiaDxfPolyline

5.8.2.14 typedef gaiaDxfText* gaiaDxfTextPtr

Typedef for DXF Text object.

See also

gaiaDxfText

5.8.3 Function Documentation

5.8.3.1 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

srid	the SRID value to be used for all Geometries
force_dims	should be one of GAIA_DXF_AUTO_2D_3D, GAIA_DXF_FORCE_2D or GAIA_DXF_FORCE_3D
prefix	an optional prefix to be used for DB target tables (could be NULL)
selected_layers	if set, only the DXF Layer of corresponding name will be imported (could be NULL)
special_rings	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().

5.8.3.2 GAIAGEO_DECLARE void gaiaDestroyDxfParser (gaiaDxfParserPtr *parser*)

Destroying a DXF Parser object.

Parameters

parser	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.

5.8.3.3 GAIAGEO_DECLARE int gaiaDxfWriteEndSection (gaiaDxfWriterPtr *dxf*)

Writing a DXF Entities Section Header.

Parameters

dxf	pointer to a properly initialized gaiaDxfWriter object

Returns

0 on failure, any other value on success

See also

gaiaDxfWriteTables, gaiaDxfWriteEntities

5.8.3.4 GAIAGEO_DECLARE int gaiaDxfWriteEntities (gaiaDxfWriterPtr *dxf*)

Writing a DXF Entities Section Header.

Parameters

dxf	pointer to a properly initialized gaiaDxfWriter object

Returns

0 on failure, any other value on success

See also

gaiaDxfWriteHeader, gaiaDxfWriteEndSection, gaiaDxfWritePoint, gaiaDxfWriteText, gaiaDxfWriteLine, gaiaDxfWriteRing, gaiaDxfWriteGeometry

5.8.3.5 GAIAGEO_DECLARE int gaiaDxfWriteFooter (gaiaDxfWriterPtr *dxf*)

Writing a DXF Entities Section Header.

Parameters

dxf	pointer to a properly initialized gaiaDxfWriter object

Returns

0 on failure, any other value on success

See also

gaiaDxfWriteHeader

5.8.3.6 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.

Parameters

dxf	pointer to a properly initialized gaiaDxfWriter object
layer_name	name of the corresponding layer
line	pointer to the internal Ring to be exported into the DXF
label	text string containing the label value (could be NULL)
text_height	only for Text Labels: ingnored in any other case.
text_rotation	only for Text Labels: ingnored in any other case.

Returns

0 on failure, any other value on success

See also

gaiaDxfWriteEntities, gaiaDxfWriteEndSection, gaiaDxfWritePoint, gaiaDxfWriteText, gaiaDxfWriteLine, gaiaDxfWriteRing

5.8.3.7 GAIAGEO_DECLARE int gaiaDxfWriteHeader (gaiaDxfWriterPtr *dxf*, double *minx*, double *miny*, double *minz*, double *maxx*, double *maxy*, double *maxz*)

Writing the DXF Header.

Parameters

dxf	pointer to a properly initialized gaiaDxfWriter object
minx	the minimum X coordinate contained within the DXF
minx	the minimum Y coordinate contained within the DXF
minx	the minimum Z coordinate contained within the DXF
minx	the maximum X coordinate contained within the DXF
minx	the maximum Y coordinate contained within the DXF
minx	the maximum Z coordinate contained within the DXF

Returns

0 on failure, any other value on success

See also

gaiaDxfWriterInit, gaiaDxfWriteFooter, gaiaDxfWriteTables, gaiaDxfWriteEntities

5.8.3.8 GAIAGEO_DECLARE int gaiaDxfWriteLayer (gaiaDxfWriterPtr *dxf*, const char * *layer_name*)

Writing a DXF Table/Layer definition.

Parameters

dxf	pointer to a properly initialized gaiaDxfWriter object
layer_name	name of the layer

Returns

0 on failure, any other value on success

See also

gaiaDxfWriteTables, gaiaDxfWriteEndSection

5.8.3.9 GAIAGEO_DECLARE int gaiaDxfWriteLine (gaiaDxfWriterPtr *dxf*, const char * *layer_name*, gaiaLinestringPtr *line*)

Writing a DXF Polyline (opened) Entity.

Parameters

dxf	pointer to a properly initialized gaiaDxfWriter object
layer_name	name of the corresponding layer
line	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

5.8.3.10 GAIAGEO_DECLARE int gaiaDxfWritePoint (gaiaDxfWriterPtr *dxf*, const char * *layer_name*, double *x*, double *y*, double *z*)

Writing a DXF Point Entity.

Parameters

dxf	pointer to a properly initialized gaiaDxfWriter object
layer_name	name of the corresponding layer
x	X coordinate value
y	Y coordinate value
z	Z coordinate value

Returns

0 on failure, any other value on success

See also

gaiaDxfWriteEntities, gaiaDxfWriteEndSection, gaiaDxfWriteText, gaiaDxfWriteLine, gaiaDxfWriteRing, gaiaDxfWriteGeometry

5.8.3.11 GAIAGEO_DECLARE int gaiaDxfWriteRing (gaiaDxfWriterPtr *dxf*, const char * *layer_name*, gaiaRingPtr *ring*)

Writing a DXF Polyline (closed) Entity.

Parameters

dxf	pointer to a properly initialized gaiaDxfWriter object
layer_name	name of the corresponding layer
line	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

5.8.3.12 GAIAGEO_DECLARE int gaiaDxfWriterInit (gaiaDxfWriterPtr *dxf*, FILE * *out*, int *precision*, int *version*)

Initializing a DXF Writer Object.

Parameters

writer	pointer to the gaiaDxfWriter object to be initialized
out	file handle to DXF output file
precision	number of decimal digits for any coordinate
version	currently always expected to be GAIA_DXF_V12

Returns

0 on failure, any other value on success

See also

gaiaDxfWriteHeader, gaiaExportDxf

5.8.3.13 GAIAGEO_DECLARE int gaiaDxfWriteTables (gaiaDxfWriterPtr *dxf*)

Writing the DXF Tables Section Header.

Parameters

dxf	pointer to a properly initialized gaiaDxfWriter object

Returns

0 on failure, any other value on success

See also

gaiaDxfWriteHeader, gaiaDxfWriteEndSection

5.8.3.14 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.

Parameters

dxf	pointer to a properly initialized gaiaDxfWriter object
layer_name	name of the corresponding layer
x	X coordinate value
y	Y coordinate value
z	Z coordinate value
label	text string containing the label value
text_height	height of the text in map units
angle	text rotation angle

Returns

0 on failure, any other value on success

See also

gaiaDxfWriteEntities, gaiaDxfWriteEndSection, gaiaDxfWritePoint, gaiaDxfWriteLine, gaiaDxfWriteRing, gaiaDxfWriteGeometry

5.8.3.15 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.

Parameters

dxf	pointer to a properly initialized gaiaDxfWriter object
db_hanlde	handle to the current DB connection
sql	a text string defining the SQL query to be used for extracting all geometries/entities to be exported into the output DXF
layer_col_name	name of the SQL resultset column containing the Layer name
geom_col_name	name of the SQL resultset column containing Geometries
label_col_name	name of the SQL resultset column containing Label values (could be NULL)
text_height_↩ col_name	name of the SQL resultset column containing Text Height values (could be NULL)
text_rotation_↩ col_name	name of the SQL resultset column containing Text Rotation values (could be NULL)
geom_filter	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

5.8.3.16 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.

Parameters

db_handle	handle to a valid DB connection
parser	pointer to DXF Parser object
mode	should be one of GAIA_DXF_IMPORT_BY_LAYER or GAIA_DXF_IMPORT_MIXED
append	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 gaiaCreateDxf↩ Parser and previously used for a succesfull call to gaiaParseDxfFile

5.8.3.17 GAIAGEO_DECLARE int gaiaParseDxfFile (gaiaDxfParserPtr *parser*, const char * *dxf_path*)

Parsing a DXF file.

Parameters

parser	pointer to DXF Parser object
dxf_path	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 gaiaCreateDxf↩ Parser. A DXF Parser object can be used only a single time to parse a DXF file.
not reentrant and thread unsafe.

5.8.3.18 GAIAGEO_DECLARE int gaiaParseDxfFile_r (const void * *p_cache*, **gaiaDxfParserPtr** *parser*, const char * *dxf_path*)

Parsing a DXF file.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
parser	pointer to DXF Parser object
dxf_path	pathname of the DXF external file to be parsed

Returns

0 on failure, any other value on success

See also

gaiaParseDxfFile, gaiaCreateDxfParser, gaiaDestroyDxfParser, gaiaLoadFromDxfParser

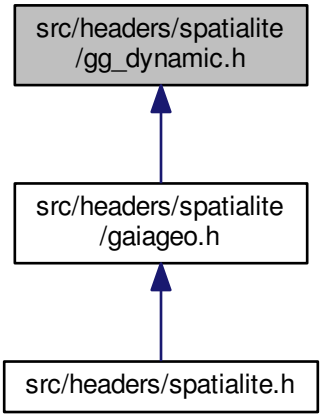
Note

the pointer to the DXF Parser object is expected to be the one returned by a previous call to gaiaCreateDxf↩ Parser. A DXF Parser object can be used only a single time to parse a DXF file.
reentrant and thread-safe.

5.9 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 dynamicly 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 dymically growing line/ring object [by coords].
- GAIAGEO_DECLARE gaiaPointPtr gaiaDynamicLineFindByPos (gaiaDynamicLinePtr p, int pos)

  Finds a Point within a dymically growing line/ring object [by position].
- GAIAGEO_DECLARE gaiaDynamicLinePtr gaiaCreateDynamicLine (double *coords, int points)

  Creates a new dynamicly growing line/ring object.

5.9.1 Detailed Description

Geometry handling functions: DynamicLine handling.

5.9.2 Function Documentation

5.9.2.1 GAIAGEO_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.

5.9.2.2 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.

Parameters

p	pointer to the dynamically growing line/ring object.
x	X coordinate of the Point
y	Y coordinate of the Point
m	M measure of the Point

Returns

the pointer to newly created Point

5.9.2.3 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.

Parameters

p	pointer to the dynamically growing line/ring object.
x	X coordinate of the Point
y	Y coordinate of the Point

Returns

the pointer to newly created Point

5.9.2.4 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.

Parameters

p	pointer to the dynamically growing line/ring object.
x	X coordinate of the Point
y	Y coordinate of the Point
z	Z coordinate of the Point
m	M measure of the Point

Returns

the pointer to newly created Point

5.9.2.5 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.

Parameters

p	pointer to the dynamically growing line/ring object.
x	X coordinate of the Point

y	Y coordinate of the Point
z	Z coordinate of the Point

Returns

the pointer to newly created Point

5.9.2.6 GAIAGEO_DECLARE gaiaDynamicLinePtr gaiaCloneDynamicLine (gaiaDynamicLinePtr *org*)

Duplicates a dynamically growing line/ring object.

Parameters

org	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.

5.9.2.7 GAIAGEO_DECLARE gaiaDynamicLinePtr gaiaCreateDynamicLine (double * *coords,* int *points*)

Creates a new dynamicly growing line/ring object.

Parameters

coords	an array of COORDs, any dimension [XY, XYZ, XYM, XYZM]
points	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.

5.9.2.8 GAIAGEO_DECLARE void gaiaDynamicLineDeletePoint (gaiaDynamicLinePtr *p,* gaiaPointPtr *pt*)

Removes a given Point from a dynamically growing line/ring object.

Parameters

p	pointer to dynamically growing line/ring object.
pt	pointer to given Point.

Note

the given Point (referenced by its address) will be removed from the dynamically growin line/ring object.
the given Point will be then implicitly destroyed.

5.9.2.9 GAIAGEO_DECLARE gaiaPointPtr gaiaDynamicLineFindByCoords (gaiaDynamicLinePtr *p*, double *x*, double *y*)

Finds a Point within a dymically growing line/ring object [by coords].

Parameters

p	pointer to dymamically line/ring object.
x	Point X coordinate.
y	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.

5.9.2.10 GAIAGEO_DECLARE gaiaPointPtr gaiaDynamicLineFindByPos (gaiaDynamicLinePtr *p*, int *pos*)

Finds a Point within a dymically growing line/ring object [by position].

Parameters

p	pointer to dymamically line/ring object.
pos	relative position [first Point has index 0].

Returns

the pointer to the corresponding Point object: NULL on failure.

See also

gaiaDynamicLineFindByCoords

5.9.2.11 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.

Parameters

p	pointer to the dynamically growing line/ring object.
pt	pointer to the given Point.
x	X coordinate of the Point to be appended
y	Y coordinate of the Point to be appended

See also

gaiaDynamicLiceInsertBefore

Returns

the pointer to newly created Point

5.9.2.12 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.

Parameters

p	pointer to the dynamically growing line/ring object.
pt	pointer to the given Point.
x	X coordinate of the Point to be appended
y	Y coordinate of the Point to be appended

See also

gaiaDynamicLiceInsertBeforeAfter

Returns

the pointer to newly created Point

5.9.2.13 GAIAGEO_DECLARE gaiaDynamicLinePtr gaiaDynamicLineJoinAfter (gaiaDynamicLinePtr *org*, gaiaPointPtr *point*, gaiaDynamicLinePtr *toJoin*)

Merges two dynamically growing line/ring object into a single one.

Parameters

org	pointer to the first input object [first line].
point	pointer to the reference Point object.
toJoin	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.

5.9.2.14 GAIAGEO_DECLARE gaiaDynamicLinePtr gaiaDynamicLineJoinBefore (gaiaDynamicLinePtr *org*, gaiaPointPtr *point*, gaiaDynamicLinePtr *toJoin*)

Merges two dynamically growing line/ring object into a single one.

Parameters

org	pointer to the first input object [first line].
point	pointer to the reference Point object.
toJoin	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.

5.9.2.15 GAIAGEO_DECLARE gaiaDynamicLinePtr gaiaDynamicLineSplitAfter (gaiaDynamicLinePtr *org*, gaiaPointPtr *point*)

Cuts a dynamically growing line/ring in two halves, using a given cut point.

Parameters

org	pointer to the input object [the line to be split].
point	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 orginal first point to the cut point [included].
on completion the orginal line will be reduced, going from the cut point [excluded] to the original last point.

5.9.2.16 GAIAGEO_DECLARE gaiaDynamicLinePtr gaiaDynamicLineSplitBefore (gaiaDynamicLinePtr *org*, gaiaPointPtr *point*)

Cuts a dynamically growing line/ring in two halves, using a given cut point.

Parameters

org	pointer to the input object [the line to be split].
point	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 orginal first point to the cut point [excluded].
on completion the orginal line will be reduced, going from the cut point [included] to the original last point.

5.9.2.17 GAIAGEO_DECLARE void gaiaFreeDynamicLine (gaiaDynamicLinePtr *p*)

Destroys a dynamically growing line/ring object.

Parameters

p	pointer to object to be destroyed

See also

gaiaAllocDynamicLine

5.9.2.18 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.

Parameters

p	pointer to the dynamically growing line/ring object.
x	X coordinate of the Point
y	Y coordinate of the Point
m	M measure of the Point

Returns

the pointer to newly created Point

5.9.2.19 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.

Parameters

p	pointer to the dynamically growing line/ring object.
x	X coordinate of the Point
y	Y coordinate of the Point

Returns

the pointer to newly created Point

5.9.2.20 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.

Parameters

p	pointer to the dynamically growing line/ring object.
x	X coordinate of the Point
y	Y coordinate of the Point
z	Z coordinate of the Point
m	M measure of the Point

Returns

the pointer to newly created Point

5.9.2.21 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.

Parameters

p	pointer to the dynamically growing line/ring object.
x	X coordinate of the Point
y	Y coordinate of the Point
z	Z coordinate of the Point

Returns

the pointer to newly created Point

5.9.2.22 GAIAGEO_DECLARE gaiaDynamicLinePtr gaiaReverseDynamicLine (gaiaDynamicLinePtr *org*)

Duplicates and reverts a dynamically growing line/ring object.

Parameters

org	pointer to dynamically growing line/ring object [origin].

Returns

the pointer to newly created dynamic growing line/ring object: NULL on failure.

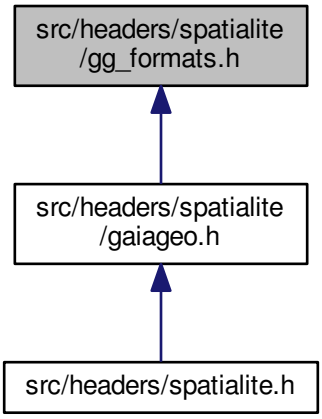
Note

the newly created object is an exact copy of the origina one, except in that direction is reverted.
i.e. first inpunt point becomes last output point, and last input point becomes first output point.

5.10 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_endian_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.
- GAIAGEO_DECLARE void gaiaMakePointZ (double x, double y, double z, int srid, unsigned char **result, int *size)

  Creates a BLOB-Geometry representing a PointZ.
- GAIAGEO_DECLARE void gaiaMakePointM (double x, double y, double m, int srid, unsigned char **result, int *size)

  Creates a BLOB-Geometry representing a PointM.
- 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.
- 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 gaiaFromSpatiaLiteBlobWkb (const unsigned char *blob, unsigned int size)

  Creates a Geometry object from the corresponding BLOB-Geometry.
- GAIAGEO_DECLARE gaiaGeomCollPtr gaiaFromSpatiaLiteBlobWkbEx (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 gaiaToSpatiaLiteBlobWkb (gaiaGeomCollPtr geom, unsigned char **result, int *size)

  Creates a BLOB-Geometry corresponding to a Geometry object.
- GAIAGEO_DECLARE void gaiaToSpatiaLiteBlobWkbEx (gaiaGeomCollPtr geom, unsigned char **result, int *size, int gpkg_mode)

  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 (gaiaOutBufferPtr out_buf, int version, int precision, gaiaGeomColl↩ Ptr 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 *charFrom, const char *charTo)

  Open a Shapefile in read mode.
- GAIAGEO_DECLARE void gaiaOpenShpWrite (gaiaShapefilePtr shp, const char *path, int shape, gaiaDbfListPtr list, const char *charFrom, const char *charTo)

  Open a Shapefile in read 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 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 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.

5.10.1 Detailed Description

Geometry handling functions: formats.

5.10.2 Function Documentation

5.10.2.1 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.

Parameters

list	pointer to the DBF List object.
name	text string: DBF Field name.
type	identifier of the corresponding DBF data type.
offset	corresponding offset into the DBF I/O buffer.
length	max field length (in bytes).
decimals	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

5.10.2.2 GAIAGEO_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.

5.10.2.3 GAIAGEO_DECLARE gaiaDbfFieldPtr gaiaAllocDbfField (char * *name*, unsigned char *type*, int *offset*, unsigned char *length*, unsigned char *decimals*)

Allocates a new DBF Field object.

Parameters

name	text string: DBF Field name.
type	identifier of the corresponding DBF data type.
offset	corresponding offset into the DBF I/O buffer.
length	max field length (in bytes).
decimals	precision: number of decimal digits.

Returns

the pointer to newly created DBF Field object.

See also

gaiaFreeDbfField, gaiaCloneDbfField, gaiaFreeValue, gaiaSetNullValue, gaiaSetIntValue, gaiaSetDouble↩Value, 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

5.10.2.4 GAIAGEO_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.

5.10.2.5 GAIAGEO_DECLARE gaiaShapefilePtr gaiaAllocShapefile (void)

Allocates a new Shapefile object.

Returns

the pointer to newly created Shapefile object.

See also

gaiaFreeShapefile, gaiaOpenShpRead, gaiaOpenShpWrite, gaiaReadShpEntity, gaiaShpAnalyze, gaia↩WriteShpEntity, gaiaFlushShpHeaders

Note

you are responsible to destroy (before or after) any allocated Shapefile.
you should phisically open the Shapefile in *read* or *write* mode before performing any actual I/O operation.

5.10.2.6 GAIAGEO_DECLARE void gaiaAppendToOutBuffer (gaiaOutBufferPtr *buf*, const char * *text*)

Appends a text string at the end of Text output buffer.

Parameters

buf	pointer to gaiaOutBufferStruct structure.
text	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.

5.10.2.7 GAIAGEO_DECLARE gaiaDbfListPtr gaiaCloneDbfEntity (gaiaDbfListPtr *org*)

Allocates a new DBF List object [duplicating an existing one].

Parameters

org	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.

5.10.2.8 GAIAGEO_DECLARE gaiaDbfFieldPtr gaiaCloneDbfField (gaiaDbfFieldPtr *org*)

Allocates a new DBF Field object [duplicating an existing one].

Parameters

org	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 evantual Field Value].

5.10.2.9 GAIAGEO_DECLARE gaiaValuePtr gaiaCloneValue (gaiaValuePtr *org*)

Allocates a new DBF Field Value object [duplicating an existing one].

Parameters

org	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.

5.10.2.10 GAIAGEO_DECLARE int gaiaEndianArch (void)

Test CPU endianness.

Returns

0 if big-endian: any other value if little-endian

5.10.2.11 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.

Parameters

geom	pointer to an existing Geometry object; if succesfull the parsed Linestring will be inserted into this Geometry
blob	pointer to EWKB input buffer
offset	the offset (in bytes) on the input buffer where the Point definition is expected to start.
blob_size	lenght (in bytes) of the input buffer.
endian	(boolean) states if the EWKB input buffer is little- or big-endian encode.
endian_arch	(boolean) states if the target CPU has a little- or big-endian architecture.
dims	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.

5.10.2.12 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.

Parameters

geom	pointer to an existing Geometry object; if succesfull the parsed MultiGeometry will be inserted into this Geometry
blob	pointer to EWKB input buffer
offset	the offset (in bytes) on the input buffer where the Point definition is expected to start.
blob_size	lenght (in bytes) of the input buffer.
endian	(boolean) states if the EWKB input buffer is little- or big-endian encode.
endian_arch	(boolean) states if the target CPU has a little- or big-endian architecture.
dims	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.

5.10.2.13 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.

Parameters

geom	pointer to an existing Geometry object; if succesfull the parsed Point will be inserted into this Geometry
blob	pointer to EWKB input buffer
offset	the offset (in bytes) on the input buffer where the Point definition is expected to start.
blob_size	lenght (in bytes) of the input buffer.
endian	(boolean) states if the EWKB input buffer is little- or big-endian encode.
endian_arch	(boolean) states if the target CPU has a little- or big-endian architecture.
dims	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.

5.10.2.14 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.

Parameters

geom	pointer to an existing Geometry object; if succesfull the parsed Polygon will be inserted into this Geometry
blob	pointer to EWKB input buffer
offset	the offset (in bytes) on the input buffer where the Point definition is expected to start.
blob_size	lenght (in bytes) of the input buffer.
endian	(boolean) states if the EWKB input buffer is little- or big-endian encode.
endian_arch	(boolean) states if the target CPU has a little- or big-endian architecture.
dims	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

5.10.2.15 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

p	endian-dependent representation (output buffer).
value	the internal value to be exported.
little_endian	0 if the output buffer has to be big-endian: any other value for little-endian.
little_endian_arch	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.

5.10.2.16 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

p	endian-dependent representation (output buffer).
value	the internal value to be exported.
little_endian	0 if the output buffer has to be big-endian: any other value for little-endian.
little_endian_arch	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.

5.10.2.17 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

p	endian-dependent representation (output buffer).
value	the internal value to be exported.
little_endian	0 if the output buffer has to be big-endian: any other value for little-endian.
little_endian_arch	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.

5.10.2.18 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

p	endian-dependent representation (output buffer).
value	the internal value to be exported.
little_endian	0 if the output buffer has to be big-endian: any other value for little-endian.
little_endian_arch	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.

5.10.2.19 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

p	endian-dependent representation (output buffer).
value	the internal value to be exported.
little_endian	0 if the output buffer has to be big-endian: any other value for little-endian.
little_endian_arch	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.

5.10.2.20 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

p	endian-dependent representation (output buffer).
value	the internal value to be exported.
little_endian	0 if the output buffer has to be big-endian: any other value for little-endian.
little_endian_arch	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.

5.10.2.21 GAIAGEO_DECLARE void gaiaFlushDbfHeader (gaiaDbfPtr *dbf*)

Writes into an output DBF File any required header / footer.

Parameters

dbf	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.

5.10.2.22 GAIAGEO_DECLARE void gaiaFlushShpHeaders (gaiaShapefilePtr *shp*)

Writes into an output Shapefile any required header / footer.

Parameters

shp	pointer to the Shapefile object.

See also

gaiaAllocShapefile, gaiaFreeShapefile, gaiaOpenShpRead, gaiaOpenShpWrite, gaiaReadShpEntity, gaia↩ ShpAnalyze, gaiaWriteShpEntity

Note

forgetting to call gaiaFlushShpHeader for any Shapefile opened in *write* mode immediately before destroying the object, will surely cause severe file corruption.

5.10.2.23 GAIAGEO_DECLARE void gaiaFreeDbf (gaiaDbfPtr *dbf*)

Destroys a DBF File object.

Parameters

dbf	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 a responsible to explicitly call gaia↩ FlushShpHeader before destroyng a Shapefile opened in *write* mode.

5.10.2.24 GAIAGEO_DECLARE void gaiaFreeDbfField (gaiaDbfFieldPtr *p*)

Destroys a DBF Field object.

Parameters

p	pointer to DBF Field object

See also

gaiaAllocDbfField, gaiaCloneDbfField, gaiaCloneValue, gaiaFreeValue, gaiaSetNullValue, gaiaSetIntValue, gaiaSetDoubleValue, gaiaSetStrValue

5.10.2.25 GAIAGEO_DECLARE void gaiaFreeDbfList (gaiaDbfListPtr *list*)

Destroys a DBF List object.

Parameters

list	pointer to the DBF List object

See also

gaiaAllocDbfList, gaiaIsValidDbfList, gaiaResetDbfEntity, gaiaCloneDbfEntity, gaiaAddDbfField

Note

attempting to destroy any DBF List object whose ownnership has already been transferred to some other (higher order) object is a serious error, and will easily cause severe memory corruption.

5.10.2.26 GAIAGEO_DECLARE void gaiaFreeShapefile (gaiaShapefilePtr *shp*)

Destroys a Shapefile object.

Parameters

shp	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 a responsible to explicitly call gaiaFlushShpHeader before destroyng a Shapefile opened in *write* mode.

5.10.2.27 GAIAGEO_DECLARE void gaiaFreeValue (gaiaValuePtr *p*)

Resets a DBF Field Value object to its initial empty state.

Parameters

p	pointer to DBF Field Value object

See also

gaiaAllocDbfField, gaiaCloneDbfField, gaiaCloneValue, gaiaSetNullValue, gaiaSetIntValue, gaiaSetDoubleValue, gaiaSetStrValue, gaiaResetDbfEntity

5.10.2.28 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaFromEWKB (const unsigned char * *in_buffer*)

Creates a Geometry object from EWKB notation.

Parameters

in_buffer	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.

5.10.2.29 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaFromFgf (const unsigned char * *blob*, unsigned int *size*)

Creates a Geometry object from FGF notation.

Parameters

blob	pointer to FGF buffer
size	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.

5.10.2.30 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaFromSpatiaLiteBlobWkb (const unsigned char * *blob*, unsigned int *size*)

Creates a Geometry object from the corresponding BLOB-Geometry.

Parameters

blob	pointer to BLOB-Geometry
size	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.

5.10.2.31 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaFromSpatiaLiteBlobWkbEx (const unsigned char * *blob*, unsigned int *size*, int *gpkg_mode*, int *gpkg_amphibious*)

Creates a Geometry object from the corresponding BLOB-Geometry.

Parameters

blob	pointer to BLOB-Geometry
size	the BLOB's size
gpkg_mode	is set to TRUE will accept only GPKG Geometry-BLOBs
gpkg_amphibious	is set to TRUE will indifferenctly accept either SpatiaLite Geometry-BLOBs or GPKG Geometry-BLOBs

Returns

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

See also

gaiaFreeGeomColl, gaiaToSpatiaLiteBlobWkb, 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.

5.10.2.32 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaFromWkb (const unsigned char * *blob*, unsigned int *size*)

Creates a Geometry object from WKB notation.

Parameters

blob	pointer to WKB buffer
size	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.

5.10.2.33 GAIAGEO_DECLARE short gaiaImport16 (const unsigned char * *p*, int *little_endian*, int *little_endian_arch*)

Import an INT-16 value in endian-aware fashion.

Parameters

p	endian-dependent representation (input buffer).
little_endian	0 if the input buffer is big-endian: any other value for little-endian.
little_endian_arch	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.

5.10.2.34 GAIAGEO_DECLARE int gaiaImport32 (const unsigned char * *p*, int *little_endian*, int *little_endian_arch*)

Import an INT-32 value in endian-aware fashion.

Parameters

p	endian-dependent representation (input buffer).
little_endian	0 if the input buffer is big-endian: any other value for little-endian.
little_endian_arch	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.

5.10.2.35 GAIAGEO_DECLARE double gaiaImport64 (const unsigned char * *p*, int *little_endian*, int *little_endian_arch*)

Import an DOUBLE-64 in endian-aware fashion.

Parameters

p	endian-dependent representation (input buffer).
little_endian	0 if the input buffer is big-endian: any other value for little-endian.
little_endian_↩ arch	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.

5.10.2.36 GAIAGEO_DECLARE float gaiaImportF32 (const unsigned char * *p*, int *little_endian*, int *little_endian_arch*)

Import a FLOAT-32 value in endian-aware fashion.

Parameters

p	endian-dependent representation (input buffer).
little_endian	0 if the input buffer is big-endian: any other value for little-endian.
little_endian_↩ arch	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.

5.10.2.37 GAIAGEO_DECLARE sqlite3_int64 gaiaImportI64 (const unsigned char * *p*, int *little_endian*, int *little_endian_arch*)

Import an INT-64 in endian-aware fashion.

Parameters

p	endian-dependent representation (input buffer).
little_endian	0 if the input buffer is big-endian: any other value for little-endian.
little_endian_↩ arch	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.

5.10.2.38 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.

Parameters

p	endian-dependent representation (input buffer).
little_endian	0 if the input buffer is big-endian: any other value for little-endian.
little_endian_arch	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.

5.10.2.39 GAIAGEO_DECLARE int gaiaIsValidDbfList (gaiaDbfListPtr *list*)

Checks a DBF List object for validity.

Parameters

list	pointer to the DBF List object.

Returns

0 if not valid: any other value if valid.

See also

gaiaAllocDbfList, gaiaFreeDbfList, gaiaIsValidDbfList, gaiaResetDbfEntity, gaiaCloneDbfEntity, gaiaAddDbfField

5.10.2.40 GAIAGEO_DECLARE void gaiaMakeLine (gaiaGeomCollPtr *geom1*, gaiaGeomCollPtr *geom2*, unsigned char ** *result*, int * *size*)

Creates a BLOB-Geometry representing a Segment (2-Points Linestring)

Parameters

geom1	pointer to first Geometry object (expected to represent a Point).
geom2	pointer to second Geometry object (expected to represent a Point).
result	on completion will containt a pointer to BLOB-Geometry: NULL on failure.
size	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].

5.10.2.41 GAIAGEO_DECLARE void gaiaMakePoint (double *x*, double *y*, int *srid*, unsigned char ** *result*, int * *size*)

Creates a BLOB-Geometry representing a Point.

Parameters

x	Point X coordinate.
y	Point Y coordinate.
srid	the SRID to be set for the Point.
result	on completion will containt a pointer to BLOB-Geometry: NULL on failure.
size	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].

5.10.2.42 GAIAGEO_DECLARE void gaiaMakePointM (double *x*, double *y*, double *m*, int *srid*, unsigned char ** *result*, int * *size*)

Creates a BLOB-Geometry representing a PointM.

Parameters

x	Point X coordinate.
y	Point Y coordinate.
m	Point M coordinate.
srid	the SRID to be set for the Point.
result	on completion will containt a pointer to BLOB-Geometry: NULL on failure.
size	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].

5.10.2.43 GAIAGEO_DECLARE void gaiaMakePointZ (double *x*, double *y*, double *z*, int *srid*, unsigned char ** *result*, int * *size*)

Creates a BLOB-Geometry representing a PointZ.

Parameters

x	Point X coordinate.
y	Point Y coordinate.
z	Point Z coordinate.
srid	the SRID to be set for the Point.
result	on completion will containt a pointer to BLOB-Geometry: NULL on failure.
size	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].

5.10.2.44 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.

Parameters

x	Point X coordinate.
y	Point Y coordinate.
z	Point Z coordinate.
m	Point M coordinate.
srid	the SRID to be set for the Point.
result	on completion will containt a pointer to BLOB-Geometry: NULL on failure.
size	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].

5.10.2.45 GAIAGEO_DECLARE void gaiaOpenDbfRead (gaiaDbfPtr *dbf*, const char * *path*, const char * *charFrom*, const char * *charTo*)

Open a DBF File in read mode.

Parameters

dbf	pointer to the DBF File object.
path	pathname to the corresponding file-system file.
charFrom	GNU ICONV name identifying the input charset encoding.
charTo	GNU ICONV name identifying the output charset encoding.

See also

gaiaAllocDbf, gaiaFreeDbf, gaiaOpenDbfWrite, 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.

5.10.2.46 GAIAGEO_DECLARE void gaiaOpenDbfWrite (gaiaDbfPtr *dbf*, const char * *path*, const char * *charFrom*, const char * *charTo*)

Open a DBF File in write mode.

Parameters

dbf	pointer to the DBF File object.
path	pathname to the corresponding file-system file.
charFrom	GNU ICONV name identifying the input charset encoding.
charTo	GNU ICONV name identifying the output charset encoding.

See also

gaiaAllocDbf, gaiaFreeDbf, gaiaOpenDbfRead, 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.

5.10.2.47 GAIAGEO_DECLARE void gaiaOpenShpRead (gaiaShapefilePtr *shp*, const char * *path*, const char * *charFrom*, const char * *charTo*)

Open a Shapefile in read mode.

Parameters

shp	pointer to the Shapefile object.
path	*abstract* pathname to the corresponding file-system files.
charFrom	GNU ICONV name identifying the input charset encoding.
charTo	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.

5.10.2.48 GAIAGEO_DECLARE void gaiaOpenShpWrite (gaiaShapefilePtr *shp*, const char * *path*, int *shape*, gaiaDbfListPtr *list*, const char * *charFrom*, const char * *charTo*)

Open a Shapefile in read mode.

Parameters

shp	pointer to the Shapefile object.
path	*abstract* pathname to the corresponding file-system files.
shape	the SHAPE code; expected to be one of GAIA_SHP_POINT, GAIA_SHP_POLYLINE, GAIA_SHP_POLYGON, GAIA_SHP_MULTIPOINT, GAIA_SHP_POINTZ, GAIA_SHP_POLYLINEZ, GAIA_SHP_POLYGONZ, GAIA_SHP_MULTIPOINTZ, GAIA_SHP_POINTM, GAIA_SHP_POLYLINEM, GAIA_SHP_POLYGONM, GAIA_SHP_MULTIPOINTM
list	pointer to DBF List object representing the corresponding data attributes.
charFrom	GNU ICONV name identifying the input charset encoding.
charTo	GNU ICONV name identifying the output charset encoding.

See also

gaiaAllocShapefile, gaiaFreeShapefile, gaiaOpenShpRead, 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.

5.10.2.49 GAIAGEO_DECLARE void gaiaOutBareKml (gaiaOutBufferPtr *out_buf*, gaiaGeomCollPtr *geom*, int *precision*)

Encodes a Geometry object into KML notation.

Parameters

out_buf	pointer to dynamically growing Text buffer
geom	pointer to Geometry object
precision	decimal digits to be used for coordinates

See also

gaiaParseKml, gaiaOutFullKml

Note

this function will export the simplest KML notation (no descriptions).

5.10.2.50 GAIAGEO_DECLARE void gaiaOutBufferInitialize (gaiaOutBufferPtr *buf*)

Initializes a dynamically growing Text output buffer.

Parameters

buf	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.

5.10.2.51 GAIAGEO_DECLARE void gaiaOutBufferReset (gaiaOutBufferPtr *buf*)

Resets a dynamically growing Text output buffer to its initial (empty) state.

Parameters

buf	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.

5.10.2.52 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

out_buf	pointer to dynamically growing Text buffer
name	text string to be set as KML *name*
desc	text string to se set as KML *description*
geom	pointer to Geometry object
precision	decimal digits to be used for coordinates

See also

gaiaParseKml, gaiaOutBareKml

Note

this function will export the simplest KML notation (no descriptions).

5.10.2.53 GAIAGEO_DECLARE void gaiaOutGeoJSON (gaiaOutBufferPtr *out_buf,* gaiaGeomCollPtr *geom,* int *precision,* int *options*)

Encodes a Geometry object into GeoJSON notation.

Parameters

out_buf	pointer to dynamically growing Text buffer
geom	pointer to Geometry object
precision	decimal digits to be used for coordinates
options	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

5.10.2.54 GAIAGEO_DECLARE void gaiaOutGml (gaiaOutBufferPtr *out_buf,* int *version,* int *precision,* gaiaGeomCollPtr *geom*)

Encodes a Geometry object into GML notation.

Parameters

out_buf	pointer to dynamically growing Text buffer
version	GML version
precision	decimal digits to be used for coordinates
geom	pointer to Geometry object

See also

gaiaParseGml

Note

if *version* is set to **3**, then GMLv3 will be used; in any other case GMLv2 will be assumed by default.

5.10.2.55 GAIAGEO_DECLARE void gaiaOutLinestringZ (gaiaOutBufferPtr *out_buf,* gaiaLinestringPtr *linestring*)

Encodes a WKT 3D Linestring [XYZ].

Parameters

out_buf	pointer to dynamically growing Text buffer
linestring	pointer to Linestring object

See also

gaiaOutPointZ, gaiaOutPolygonZ, gaiaOutLinestringZex

Remarks

mainly intended for internal usage.

5.10.2.56 GAIAGEO_DECLARE void gaiaOutLinestringZex (gaiaOutBufferPtr *out_buf*, gaiaLinestringPtr *linestring*, int *precision*)

Encodes a WKT 3D Linestring [XYZ].

Parameters

out_buf	pointer to dynamically growing Text buffer
linestring	pointer to Linestring object
precision	decimal digits to be used for coordinates

See also

gaiaOutPointZ, gaiaOutPolygonZ

Remarks

mainly intended for internal usage.

5.10.2.57 GAIAGEO_DECLARE void gaiaOutPointZ (gaiaOutBufferPtr *out_buf*, gaiaPointPtr *point*)

Encodes a WKT 3D Point [XYZ].

Parameters

out_buf	pointer to dynamically growing Text buffer
point	pointer to Point object

See also

gaiaOutLinestringZ, gaiaOutPolygonZ, gaiaOutPointZex

Remarks

mainly intended for internal usage.

5.10.2.58 GAIAGEO_DECLARE void gaiaOutPointZex (gaiaOutBufferPtr *out_buf*, gaiaPointPtr *point*, int *precision*)

Encodes a WKT 3D Point [XYZ].

Parameters

out_buf	pointer to dynamically growing Text buffer
point	pointer to Point object
precision	decimal digits to be used for coordinates

See also

gaiaOutLinestringZ, gaiaOutPolygonZ

Remarks

mainly intended for internal usage.

5.10.2.59 GAIAGEO_DECLARE void gaiaOutPolygonZ (gaiaOutBufferPtr *out_buf*, gaiaPolygonPtr *polygon*)

Encodes a WKT 3D Polygon [XYZ].

Parameters

out_buf	pointer to dynamically growing Text buffer
polygon	pointer to Point object

See also

gaiaOutPointZ, gaiaOutLinestringZ, gaiaOutPolygonZex

Remarks

mainly intended for internal usage.

5.10.2.60 GAIAGEO_DECLARE void gaiaOutPolygonZex (gaiaOutBufferPtr *out_buf,* gaiaPolygonPtr *polygon,* int *precision*)

Encodes a WKT 3D Polygon [XYZ].

Parameters

out_buf	pointer to dynamically growing Text buffer
polygon	pointer to Point object
precision	decimal digits to be used for coordinates

See also

gaiaOutPointZ, gaiaOutLinestringZ

Remarks

mainly intended for internal usage.

5.10.2.61 GAIAGEO_DECLARE void gaiaOutSvg (gaiaOutBufferPtr *out_buf,* gaiaGeomCollPtr *geom,* int *relative,* int *precision*)

Encodes a Geometry object into SVG notation.

Parameters

out_buf	pointer to dynamically growing Text buffer
geom	pointer to Geometry object
relative	flag: relative or absolute coordinates
precision	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.

5.10.2.62 GAIAGEO_DECLARE void gaiaOutWkt (gaiaOutBufferPtr *out_buf,* gaiaGeomCollPtr *geom*)

Encodes a Geometry object into WKT notation.

Parameters

out_buf	pointer to dynamically growing Text buffer
geom	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.

5.10.2.63 GAIAGEO_DECLARE void gaiaOutWktEx (gaiaOutBufferPtr *out_buf*, gaiaGeomCollPtr *geom*, int *precision*)

Encodes a Geometry object into WKT notation.

Parameters

out_buf	pointer to dynamically growing Text buffer
geom	pointer to Geometry object
precision	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.

5.10.2.64 GAIAGEO_DECLARE void gaiaOutWktStrict (gaiaOutBufferPtr *out_buf*, gaiaGeomCollPtr *geom*, int *precision*)

Encodes a Geometry object into strict 2D WKT notation.

Parameters

out_buf	pointer to dynamically growing Text buffer
geom	pointer to Geometry object
precision	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.

5.10.2.65 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaParseEWKT (const unsigned char * *in_buffer*)

Creates a Geometry object from EWKT notation.

Parameters

in_buffer	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.

5.10.2.66 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaParseGeoJSON (const unsigned char * *in_buffer*)

Creates a Geometry object from GeoJSON notation.

Parameters

in_buffer	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.

5.10.2.67 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaParseGml (const unsigned char * *in_buffer*, sqlite3 * *sqlite_handle*)

Creates a Geometry object from GML notation.

Parameters

in_buffer	pointer to GML buffer
sqlite_handle	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.

5.10.2.68 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaParseGml_r (const void * *p_cache*, const unsigned char * *in_buffer*, sqlite3 * *sqlite_handle*)

Creates a Geometry object from GML notation.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
in_buffer	pointer to GML buffer
sqlite_handle	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.

5.10.2.69 GAIAGEO_DECLARE unsigned char* gaiaParseHexEWKB (const unsigned char * *blob_hex*, int * *blob_size*)

Translates an EWKB notation from hexadecimal into binary.

Parameters

blob_hex	pointer to EWKB input buffer (hexadecimal text string)
blob_size	lenght (in bytes) of the input buffer; if succesfull will contain the lenght 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()

5.10.2.70 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaParseKml (const unsigned char * *in_buffer*)

Creates a Geometry object from KML notation.

Parameters

in_buffer	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.

5.10.2.71 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaParseWkt (const unsigned char * *in_buffer*, short *type*)

Creates a Geometry object from WKT notation.

Parameters

in_buffer	pointer to WKT buffer
type	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.

5.10.2.72 GAIAGEO_DECLARE int gaiaReadDbfEntity (gaiaDbfPtr *dbf*, int *current_row*, int * *deleted*)

Reads a record from a DBF File object.

Parameters

dbf	pointer to the DBF File object.
current_row	the row number identifying the record to be read.
deleted	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 *logically deleted*.

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.

5.10.2.73 GAIAGEO_DECLARE int gaiaReadDbfEntity_ex (gaiaDbfPtr *dbf*, int *current_row*, int * *deleted*, int *text_dates*)

Reads a record from a DBF File object.

Parameters

dbf	pointer to the DBF File object.
current_row	the row number identifying the record to be read.
deleted	on completion this variable will contain 0 if the record
text_dates	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 *logically deleted*.

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.

5.10.2.74 GAIAGEO_DECLARE int gaiaReadShpEntity (gaiaShapefilePtr *shp*, int *current_row*, int *srid*)

Reads a feature from a Shapefile object.

Parameters

shp	pointer to the Shapefile object.
current_row	the row number identifying the feature to be read.
srid	feature's SRID

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.

5.10.2.75 GAIAGEO_DECLARE int gaiaReadShpEntity_ex (gaiaShapefilePtr *shp*, int *current_row*, int *srid*, int *text_dates*)

Reads a feature from a Shapefile object.

Parameters

shp	pointer to the Shapefile object.
current_row	the row number identifying the feature to be read.
srid	feature's SRID
text_dates	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.

5.10.2.76 GAIAGEO_DECLARE void gaiaResetDbfEntity (gaiaDbfListPtr *list*)

Resets a DBF List object to its initial empty state.

Parameters

list	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).

5.10.2.77 GAIAGEO_DECLARE void gaiaSetDoubleValue (gaiaDbfFieldPtr *field*, double *value*)

Sets a DOUBLE current value for a DBF Field object.

Parameters

field	pointer to DBF Field object.
value	double value to be set.

See also

gaiaAllocDbfField, gaiaFreeDbfField, gaiaCloneDbfField, gaiaFreeValue, gaiaSetNullValue, gaiaSetIntValue, gaiaSetStrValue

5.10.2.78 GAIAGEO_DECLARE void gaiaSetIntValue (gaiaDbfFieldPtr *field*, sqlite3_int64 *value*)

Sets an INTEGER current value for a DBF Field object.

Parameters

field	pointer to DBF Field object.
value	integer value to be set.

See also

gaiaAllocDbfField, gaiaFreeDbfField, gaiaCloneDbfField, gaiaFreeValue, gaiaSetNullValue, gaiaSetDouble↩Value, gaiaSetStrValue

5.10.2.79 GAIAGEO_DECLARE void gaiaSetNullValue (gaiaDbfFieldPtr *field*)

Sets a NULL current value for a DBF Field object.

Parameters

field	pointer to DBF Field object

See also

gaiaAllocDbfField, gaiaFreeDbfField, gaiaCloneDbfField, gaiaFreeValue, gaiaSetIntValue, gaiaSetDouble↩Value, gaiaSetStrValue

5.10.2.80 GAIAGEO_DECLARE void gaiaSetStrValue (gaiaDbfFieldPtr *field*, char * *str*)

Sets a TEXT current value for a DBF Field object.

Parameters

field	pointer to DBF Field object.
str	text string value to be set.

See also

gaiaAllocDbfField, gaiaFreeDbfField, gaiaCloneDbfField, gaiaFreeValue, gaiaSetNullValue, gaiaSetIntValue, gaiaSetDoubleValue

5.10.2.81 GAIAGEO_DECLARE void gaiaShpAnalyze (gaiaShapefilePtr *shp*)

Prescans a Shapefile object gathering informations.

Parameters

shp	pointer to the Shapefile object.

See also

gaiaAllocShapefile, gaiaFreeShapefile, gaiaOpenShpRead, gaiaOpenShpWrite, gaiaReadShpEntity, gaiaWriteShpEntity, gaiaFlushShpHeaders

Note

on completion the Shapefile's *EffectiveType* will containt the Geometry type corresponding to features actually found.

Remarks

the Shapefile object should be opened in *read* mode.

5.10.2.82 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.

Parameters

path	to the corresponding file-system file.
field_separator	the character acting as a separator between adjacent fields.
text_separator	the character used to quote text strings.
decimal_separator	the character used as a separator between integer and decimal digits for real numeric values.
first_line_titles	0 if the first line contains regular values: any other value if the first line contains column names.
encoding	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.

5.10.2.83 GAIAGEO_DECLARE void gaiaTextReaderDestroy (gaiaTextReaderPtr *reader*)

Destroys a Text Reader object.

Parameters

reader	pointer to Text Reader object.

See also

gaiaTextReaderAlloc, gaiaTextReaderParse, gaiaTextReaderGetRow, gaiaTextReaderFetchField

5.10.2.84 GAIAGEO_DECLARE int gaiaTextReaderFetchField (gaiaTextReaderPtr *reader*, int *field_num*, int * *type*, const char ** *value*)

Retrieves an individual field value from the current Line.

Parameters

reader	pointer to Text Reader object.
field_num	relative field [aka column] index: first field has index 0.
type	on completion this variable will contain the value type.
value	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

5.10.2.85 GAIAGEO_DECLARE int gaiaTextReaderGetRow (gaiaTextReaderPtr *reader*, int *row_num*)

Reads a line from a Text Reader object.

Parameters

reader	pointer to Text Reader object.
row_num	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.

5.10.2.86 GAIAGEO_DECLARE int gaiaTextReaderParse (gaiaTextReaderPtr *reader*)

Prescans the external file associated to a Text Reade object.

Parameters

reader	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.

5.10.2.87 GAIAGEO_DECLARE void gaiaToCompressedBlobWkb (gaiaGeomCollPtr *geom*, unsigned char ** *result*, int * *size*)

Creates a Compressed BLOB-Geometry corresponding to a Geometry object.

Parameters

geom	pointer to the Geometry object.
result	on completion will containt a pointer to Compressed BLOB-Geometry: NULL on failure.
size	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].

5.10.2.88 GAIAGEO_DECLARE void gaiaToEWKB (gaiaOutBufferPtr *out_buf*, gaiaGeomCollPtr *geom*)

Encodes a Geometry object into EWKB notation.

Parameters

out_buf	pointer to dynamically growing Text buffer
geom	pointer to Geometry object

See also

gaiaFromWkb, gaiaToWkb, gaiaToHexWkb, gaiaFromEWKB, gaiaToEWKB

Note

this function will produce strictly conformat EWKB; you can safely use this for PostGIS data exchange.

5.10.2.89 GAIAGEO_DECLARE void gaiaToEWKT (gaiaOutBufferPtr *out_buf,* gaiaGeomCollPtr *geom*)

Encodes a Geometry object into EWKT notation.

Parameters

out_buf	pointer to dynamically growing Text buffer
geom	pointer to Geometry object

See also

gaiaParseWkt, gaiaOutWkt, gaiaOutWktStrict, gaiaParseEWKT

Note

this function will apply PostGIS own EWKT encoding.

5.10.2.90 GAIAGEO_DECLARE void gaiaToFgf (gaiaGeomCollPtr *geom*, unsigned char ** *result*, int * *size*, int *coord_dims*)

Encodes a Geometry object into FGF notation.

Parameters

geom	pointer to Geometry object
result	on completion will containt a pointer to the FGF buffer [BLOB]: NULL on failure.
size	on completion this variable will contain the BLOB's size (in bytes)
coord_dims	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].

5.10.2.91 GAIAGEO_DECLARE char* gaiaToHexWkb (gaiaGeomCollPtr *geom*)

Encodes a Geometry object into (hex) WKB notation.

Parameters

geom	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].

5.10.2.92 GAIAGEO_DECLARE void gaiaToSpatiaLiteBlobWkb (gaiaGeomCollPtr *geom*, unsigned char ** *result*, int * *size*)

Creates a BLOB-Geometry corresponding to a Geometry object.

Parameters

geom	pointer to the Geometry object.
result	on completion will containt a pointer to BLOB-Geometry: NULL on failure.
size	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.

5.10.2.93 GAIAGEO_DECLARE void gaiaToSpatiaLiteBlobWkbEx (gaiaGeomCollPtr *geom*, unsigned char ** *result*, int * *size*, int *gpkg_mode*)

Creates a BLOB-Geometry corresponding to a Geometry object.

Parameters

geom	pointer to the Geometry object.
result	on completion will containt a pointer to BLOB-Geometry: NULL on failure.
size	on completion this variable will contain the BLOB's size (in bytes)
gpkg_mode	is set to TRUE will always return GPKG Geometry-BLOBs

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].

5.10.2.94 GAIAGEO_DECLARE void gaiaToWkb (gaiaGeomCollPtr *geom*, unsigned char ** *result*, int * *size*)

Encodes a Geometry object into WKB notation.

Parameters

geom	pointer to Geometry object
result	on completion will containt a pointer to the WKB buffer [BLOB]: NULL on failure.
size	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].

5.10.2.95 GAIAGEO_DECLARE int gaiaWriteDbfEntity (gaiaDbfPtr *dbf*, gaiaDbfListPtr *entity*)

Writes a record into a DBF File object.

Parameters

dbf	pointer to the DBF File object.
entity	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.

5.10.2.96 GAIAGEO_DECLARE int gaiaWriteShpEntity (gaiaShapefilePtr *shp*, gaiaDbfListPtr *entity*)

Writes a feature into a Shapefile object.

Parameters

shp	pointer to the Shapefile object.
entity	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, gaia↩ShpAnalyze, gaiaFlushShpHeaders

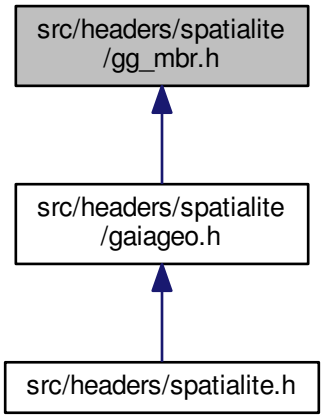
Remarks

the Shapefile object should be opened in *write* mode.

5.11 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 gaiaZRangeRing (gaiaRingPtr rng, double *min, double *max)

  Computes the Z-Range for a Ring object.
- GAIAGEO_DECLARE void gaiaZRangePolygon (gaiaPolygonPtr polyg, double *min, double *max)

  Computes the Z-Range for a Polygon object.
- GAIAGEO_DECLARE void gaiaZRangeGeometry (gaiaGeomCollPtr geom, double *min, double *max)

  Computes the Z-Range for a Geometry object.
- GAIAGEO_DECLARE void gaiaMRangeLinestring (gaiaLinestringPtr line, double *min, double *max)

  Computes the M-Range for a Linestring object.
- GAIAGEO_DECLARE void gaiaMRangeRing (gaiaRingPtr rng, double *min, double *max)

  Computes the M-Range for a Ring object.
- GAIAGEO_DECLARE void gaiaMRangePolygon (gaiaPolygonPtr polyg, double *min, double *max)

  Computes the M-Range for a Polygon object.
- GAIAGEO_DECLARE void gaiaMRangeGeometry (gaiaGeomCollPtr geom, double *min, double *max)

  Computes the Z-Range for a Geometry object.

5.11.1 Detailed Description

Geometry handling functions: MBR.

5.11.2 Function Documentation

5.11.2.1 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

x	centre X coordinate.

y	centre Y coordinate.
radius	the radius of the circle
srid	the SRID associated to the Envelope
result	on completion will contain a pointer to newly created BLOB-Geometry
size	on completion this variabile will contain the BLOB's size (in bytes)

See also

gaiaBuildMbr

Note

the *circle* of givern *radius* and *centre* will be used so to determine the corresponding *square* Envelope

5.11.2.2 GAIAGEO_DECLARE void gaiaBuildFilterMbr (double *x1*, double *y1*, double *x2*, double *y2*, int *mode*, unsigned char ** *result*, int * *size*)

Creates a BLOB-FilterMBR.

Parameters

x1	first X coordinate.
y1	first Y coordinate.
x2	second X coordinate.
y2	second Y coordinate.
mode	one of: GAIA_FILTER_MBR_WITHIN, GAIA_FILTER_MBR_CONTAINS, GAIA_FILTER_↩ MBR_INTERSECTS, GAIA_FILTER_MBR_DECLARE
result	on completion will contain a pointer to newly created BLOB-FilterMBR
size	on completion this variabile 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.

5.11.2.3 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].

Parameters

x1	first X coordinate.

y1	first Y coordinate.
x2	second X coordinate.
y2	second Y coordinate.
srid	the SRID associated to the Envelope
result	on completion will contain a pointer to newly created BLOB-Geometry
size	on completion this variabile 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.

5.11.2.4 GAIAGEO_DECLARE gaiaGeomCollPtr gaiaFromSpatiaLiteBlobMbr (const unsigned char * *blob*, unsigned int *size*)

Creates a Geometry object corresponding to the Envelope [MBR] for a BLOB-Geometry.

Parameters

blob	pointer to BLOB-Geometry
size	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.

5.11.2.5 GAIAGEO_DECLARE int gaiaGetMbrMaxX (const unsigned char * *blob*, unsigned int *size*, double * *maxx*)

Retrieves the MBR (MaxX) from a BLOB-Geometry object.

Parameters

blob	pointer to BLOB-Geometry.
size	the BLOB's size (in bytes).
maxx	on completion this variable will contain the MBR MaxX coordinate.

Returns

0 on failure: any other value on success.

See also

gaiaGetMbrMinX, gaiaGetMbrMinY, gaiaGetMbrMaxY

5.11.2.6 GAIAGEO_DECLARE int gaiaGetMbrMaxY (const unsigned char * *blob,* unsigned int *size,* double * *maxy*)

Retrieves the MBR (MaxY) from a BLOB-Geometry object.

Parameters

blob	pointer to BLOB-Geometry.
size	the BLOB's size (in bytes).
maxy	on completion this variable will contain the MBR MaxY coordinate.

Returns

0 on failure: any other value on success.

See also

gaiaGetMbrMinX, gaiaGetMbrMaxX, gaiaGetMbrMinY

5.11.2.7 GAIAGEO_DECLARE int gaiaGetMbrMinX (const unsigned char * *blob*, unsigned int *size*, double * *minx*)

Retrieves the MBR (MinX) from a BLOB-Geometry object.

Parameters

blob	pointer to BLOB-Geometry.
size	the BLOB's size (in bytes).
minx	on completion this variable will contain the MBR MinX coordinate.

Returns

0 on failure: any other value on success.

See also

gaiaGetMbrMaxX, gaiaGetMbrMinY, gaiaGetMbrMaxY

5.11.2.8 GAIAGEO_DECLARE int gaiaGetMbrMinY (const unsigned char * *blob*, unsigned int *size*, double * *miny*)

Retrieves the MBR (MinY) from a BLOB-Geometry object.

Parameters

blob	pointer to BLOB-Geometry.
size	the BLOB's size (in bytes).
miny	on completion this variable will contain the MBR MinY coordinate.

Returns

0 on failure: any other value on success.

See also

gaiaGetMbrMinX, gaiaGetMbrMaxX, gaiaGetMbrMaxY

5.11.2.9 GAIAGEO_DECLARE void gaiaMbrGeometry (gaiaGeomCollPtr *geom*)

Updates the actual MBR for a Geometry object.

Parameters

geom	pointer to the Geometry object

5.11.2.10 GAIAGEO_DECLARE void gaiaMbrLinestring (gaiaLinestringPtr *line*)

Updates the actual MBR for a Linestring object.

Parameters

line	pointer to the Linestring object

5.11.2.11 GAIAGEO_DECLARE void gaiaMbrPolygon (gaiaPolygonPtr *polyg*)

Updates the actual MBR for a Polygon object.

Parameters

polyg	pointer to the Polygon object

5.11.2.12 GAIAGEO_DECLARE void gaiaMbrRing (gaiaRingPtr *rng*)

Updates the actual MBR for a Ring object.

Parameters

rng	pointer to the Ring object

5.11.2.13 GAIAGEO_DECLARE int gaiaMbrsContains (gaiaGeomCollPtr *mbr1*, gaiaGeomCollPtr *mbr2*)

MBRs comparison: Contains.

Parameters

mbr1	pointer to first Geometry object.
mbr2	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

5.11.2.14 GAIAGEO_DECLARE int gaiaMbrsDisjoint (gaiaGeomCollPtr *mbr1*, gaiaGeomCollPtr *mbr2*)

MBRs comparison: Disjoint.

Parameters

mbr1	pointer to first Geometry object.
mbr2	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

5.11.2.15 GAIAGEO_DECLARE int gaiaMbrsEqual (gaiaGeomCollPtr *mbr1*, gaiaGeomCollPtr *mbr2*)

MBRs comparison: Equal.

Parameters

mbr1	pointer to first Geometry object.
mbr2	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

5.11.2.16 GAIAGEO_DECLARE int gaiaMbrsIntersects (gaiaGeomCollPtr *mbr1*, gaiaGeomCollPtr *mbr2*)

MBRs comparison: Intersects.

Parameters

mbr1	pointer to first Geometry object.
mbr2	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

5.11.2.17 GAIAGEO_DECLARE int gaiaMbrsOverlaps (gaiaGeomCollPtr *mbr1*, gaiaGeomCollPtr *mbr2*)

MBRs comparison: Overlaps.

Parameters

mbr1	pointer to first Geometry object.
mbr2	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

5.11.2.18 GAIAGEO_DECLARE int gaiaMbrsTouches (gaiaGeomCollPtr *mbr1,* gaiaGeomCollPtr *mbr2*)

MBRs comparison: Touches.

Parameters

mbr1	pointer to first Geometry object.
mbr2	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

5.11.2.19 GAIAGEO_DECLARE int gaiaMbrsWithin (gaiaGeomCollPtr *mbr1,* gaiaGeomCollPtr *mbr2*)

MBRs comparison: Within.

Parameters

mbr1	pointer to first Geometry object.
mbr2	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, gaiaMbrs↩Touches

5.11.2.20 GAIAGEO_DECLARE void gaiaMRangeGeometry (gaiaGeomCollPtr *geom,* double * *min,* double * *max*)

Computes the Z-Range for a Geometry object.

Parameters

geom	pointer to the Geometry object
min	on completion this variable will contain the min M value found
max	on completion this variable will contain the max M value found

Note

if the Geometry has XY or XYZ dims, the M-Range is meaningless

5.11.2.21 GAIAGEO_DECLARE void gaiaMRangeLinestring (gaiaLinestringPtr *line*, double * *min*, double * *max*)

Computes the M-Range for a Linestring object.

Parameters

line	pointer to the Linestring object
min	on completion this variable will contain the min M value found
max	on completion this variable will contain the max M value found

Note

if the Linestring has XY or XYZ dims, the M-Range is meaningless

5.11.2.22 GAIAGEO_DECLARE void gaiaMRangePolygon (gaiaPolygonPtr *polyg*, double * *min*, double * *max*)

Computes the M-Range for a Polygon object.

Parameters

polyg	pointer to the Polygon object
min	on completion this variable will contain the min M value found
max	on completion this variable will contain the max M value found

Note

if the Polygon has XY or XYZ dims, the M-Range is meaningless

5.11.2.23 GAIAGEO_DECLARE void gaiaMRangeRing (gaiaRingPtr *rng*, double * *min*, double * *max*)

Computes the M-Range for a Ring object.

Parameters

rng	pointer to the Ring object
min	on completion this variable will contain the min M value found
max	on completion this variable will contain the max M value found

Note

if the Ring has XY or XYZ dims, the M-Range is meaningless

5.11.2.24 GAIAGEO_DECLARE int gaiaParseFilterMbr (unsigned char * *result*, int *size*, double * *minx*, double * *miny*, double * *maxx*, double * *maxy*, int * *mode*)

Creates a BLOB-FilterMBR.

Parameters

result	pointer to BLOB-FilterMBR [previously created by gaiaBuildFilterMbr] BLOB-Geometry
size	BLOB's size (in bytes)
minx	on completion this variable will contain the MBR MinX coord.
miny	on completion this variable will contain the MBR MinY coord.
maxx	on completion this variable will contain the MBR MinY coord.
maxy	on completion this variable will contain the MBR MaxY coord.
mode	on completion this variable will contain the FilterMBR mode.

See also

gaiaBuildFilterMbr

Remarks

internally used to implement Geometry Callback R∗Tree filtering.

5.11.2.25 GAIAGEO_DECLARE void gaiaZRangeGeometry (gaiaGeomCollPtr *geom*, double ∗ *min*, double ∗ *max*)

Computes the Z-Range for a Geometry object.

Parameters

geom	pointer to the Geometry object
min	on completion this variable will contain the min Z value found
max	on completion this variable will contain the max Z value found

Note

if the Geometry has XY or XYM dims, the Z-Range is meaningless

5.11.2.26 GAIAGEO_DECLARE void gaiaZRangeLinestring (gaiaLinestringPtr *line*, double ∗ *min*, double ∗ *max*)

Computes the Z-Range for a Linestring object.

Parameters

line	pointer to the Linestring object
min	on completion this variable will contain the min Z value found
max	on completion this variable will contain the max Z value found

Note

if the Linestring has XY or XYM dims, the Z-Range is meaningless

5.11.2.27 GAIAGEO_DECLARE void gaiaZRangePolygon (gaiaPolygonPtr *polyg*, double ∗ *min*, double ∗ *max*)

Computes the Z-Range for a Polygon object.

Parameters

polyg	pointer to the Polygon object
min	on completion this variable will contain the min Z value found
max	on completion this variable will contain the max Z value found

Note

if the Polygon has XY or XYM dims, the Z-Range is meaningless

5.11.2.28 GAIAGEO_DECLARE void gaiaZRangeRing (gaiaRingPtr *rng*, double * *min*, double * *max*)

Computes the Z-Range for a Ring object.

Parameters

rng	pointer to the Ring object
min	on completion this variable will contain the min Z value found
max	on completion this variable will contain the max Z value found

Note

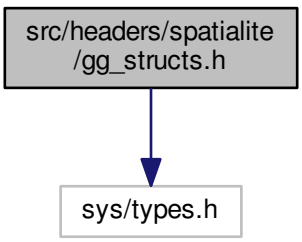
if the Ring has XY or XYM dims, the Z-Range is meaningless

5.12 src/headers/spatialite/gg_structs.h File Reference

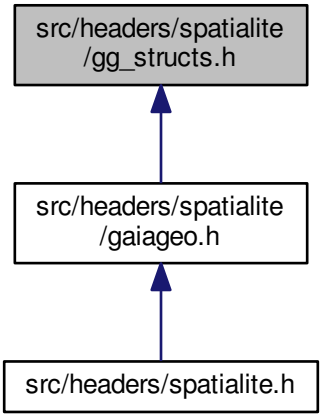
Geometry structures.

```
#include <sys/types.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 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.

Macros

- #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 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.

5.12.1 Detailed Description

Geometry structures.

5.12.2 Typedef Documentation

5.12.2.1 typedef gaiaAttributeFieldDoubleRange* gaiaAttributeFieldDoubleRangePtr

Typedef for Attribute/Field Double range infos.

See also

gaiaAttributeFieldDoubleRange

5.12.2.2 typedef gaiaAttributeFieldIntRange* gaiaAttributeFieldIntRangePtr

Typedef for Attribute/Field Integer range infos.

See also

gaiaAttributeFieldIntRange

5.12.2.3 typedef gaiaAttributeFieldMaxSize* gaiaAttributeFieldMaxSizePtr

Typedef for Attribute/Field MaxSize/Length infos.

See also

gaiaAttributeFieldMaxSize

5.12.2.4 typedef gaiaDbfList* gaiaDbfListPtr

Typedef for a list of DBF fields.

See also

gaiaDbfList

5.12.2.5 typedef gaiaDbf* gaiaDbfPtr

Typedef for DBF file handler structure.

See also

gaiaDbf

5.12.2.6 typedef gaiaDynamicLine* gaiaDynamicLinePtr

Typedef for dynamically growing line/ring structure.

See also

gaiaDynamicLine

5.12.2.7 typedef gaiaGeomColl* gaiaGeomCollPtr

Typedef for OGC GEOMETRYCOLLECTION structure.

See also

gaiaGeomCool

5.12.2.8 typedef gaiaLayerAttributeField* gaiaLayerAttributeFieldPtr

Typedef for Layer AttributeField infos.

See also

gaiaLayerAttributeField

5.12.2.9 typedef gaiaLayerAuth* gaiaLayerAuthPtr

Typedef for Layer Auth infos.

See also

gaiaLayerAuth

5.12.2.10 typedef gaiaLayerExtent* gaiaLayerExtentPtr

Typedef for Layer Extent infos.

See also

gaiaLayerExtent

5.12.2.11 typedef gaiaLinestring* gaiaLinestringPtr

Typedef for OGC LINESTRING structure.

See also

gaiaLinestring

5.12.2.12 typedef gaiaOutBuffer* gaiaOutBufferPtr

Typedef for dynamically growing output buffer structure.

See also

gaiaOutBuffer

5.12.2.13 typedef gaiaPoint* gaiaPointPtr

Typedef for OGC POINT structure.

See also

gaiaPoint

5.12.2.14 typedef gaiaPolygon* gaiaPolygonPtr

Typedef for OGC POLYGON structure.

See also

gaiaPolygon

5.12.2.15 typedef gaiaPreRing* gaiaPreRingPtr

Typedef for gaiaPreRing structure.

See also

gaiaPreRing

5.12.2.16 typedef gaiaRing* gaiaRingPtr

Typedef for OGC RING structure.

See also

gaiaRing

5.12.2.17 typedef gaiaShapefile* gaiaShapefilePtr

Typedef for SHP file handler structure.

See also

gaiaShapefile

5.12.2.18 typedef gaiaTextReader* gaiaTextReaderPtr

Typedef for Virtual Text file handling structure.

See also

gaiaTextReader

5.12.2.19 typedef gaiaVectorLayer* gaiaVectorLayerPtr

Typedef for Vector Layer item.

See also

gaiaVectorLayer

5.12.2.20 typedef gaiaVectorLayersList* gaiaVectorLayersListPtr

Typedef for Vector Layers List.

See also

gaiaVectorLayersList

5.13 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 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.

5.13.1 Detailed Description

WFS support.

5.13.2 Function Documentation

5.13.2.1 SPATIALITE_DECLARE gaiaWFScatalogPtr create_wfs_catalog (const char * *path_or_url*, char ** *err_msg*)

Creates a Catalog for some WFS service.

Parameters

path_or_url	pointer to some WFS-GetCapabilities XML Document (could be a pathname or an URL).
err_msg	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().

5.13.2.2 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

path_or_url	pointer to some WFS-DescribeFeatureType XML Document (could be a pathname or an URL).

err_msg	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().

5.13.2.3 SPATIALITE_DECLARE void destroy_wfs_catalog (gaiaWFScatalogPtr *handle*)

Destroys a WFS-Catalog object freeing any allocated resource.

Parameters

handle	the pointer to a valid WFS-Catalog returned by a previous call to create_wfs_catalog()

See also

create_wfs_catalog

5.13.2.4 SPATIALITE_DECLARE void destroy_wfs_schema (gaiaWFSschemaPtr *handle*)

Destroys a WFS-schema object freeing any allocated resource.

Parameters

handle	the pointer to a valid WFS-Catalog returned by a previous call to create_wfs_schema()

See also

create_wfs_schema

5.13.2.5 SPATIALITE_DECLARE const char* get_wfs_base_describe_url (gaiaWFScatalogPtr *handle*)

Return the base URL for any WFS-DescribeFeatureType call.

Parameters

handle	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

5.13.2.6 SPATIALITE_DECLARE const char* get_wfs_base_request_url (gaiaWFScatalogPtr *handle*)

Return the base URL for any WFS-GetFeature call.

Parameters

handle	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

5.13.2.7 SPATIALITE_DECLARE int get_wfs_catalog_count (gaiaWFScatalogPtr *handle*)

Return the total count of items (aka Layers) defined within a WFS-Catalog object.

Parameters

handle	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

5.13.2.8 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

handle	the pointer to a valid WFS-Catalog returned by a previous call to create_wfs_catalog()
index	the relative index identifying the required WFS-Layer (the first Item in the WFS-Catalaog 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

5.13.2.9 SPATIALITE_DECLARE char* get_wfs_describe_url (gaiaWFScatalogPtr *handle,* const char * *name,* const char * *version*)

Return a DescribeFeatureType URL (GET)

Parameters

handle	the pointer to a valid WFS-Item returned by a previous call to get_wfs_catalog_item().
name	the NAME uniquely identifying the required WFS layer.
version	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().

5.13.2.10 SPATIALITE_DECLARE const char* get_wfs_item_abstract (gaiaWFSitemPtr *handle*)

Return the abstract corresponding to some WFS-Item (aka Layer) object.

Parameters

handle	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

5.13.2.11 SPATIALITE_DECLARE const char* get_wfs_item_name (gaiaWFSitemPtr *handle*)

Return the name corresponding to some WFS-Item (aka Layer) object.

Parameters

handle	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

5.13.2.12 SPATIALITE_DECLARE const char* get_wfs_item_title (gaiaWFSitemPtr *handle*)

Return the title corresponding to some WFS-Item (aka Layer) object.

Parameters

handle	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

5.13.2.13 SPATIALITE_DECLARE const char* get_wfs_keyword (gaiaWFSitemPtr *handle*, int *index*)

Return one of the Keywords supported by a WFS-Item object.

Parameters

handle	the pointer to a valid WFS-Item returned by a previous call to get_wfs_catalog_item().
index	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

5.13.2.14 SPATIALITE_DECLARE int get_wfs_keyword_count (gaiaWFSitemPtr *handle*)

Return the total count of Keywords associated to a WFS-Item object.

Parameters

handle	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

5.13.2.15 SPATIALITE_DECLARE int get_wfs_layer_srid (gaiaWFSitemPtr *handle*, int *index*)

Return one of the SRIDs supported by a WFS-Item object.

Parameters

handle	the pointer to a valid WFS-Item returned by a previous call to get_wfs_catalog_item().
index	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

5.13.2.16 SPATIALITE_DECLARE int get_wfs_layer_srid_count (gaiaWFSitemPtr *handle*)

Return the total count of SRIDs supported by a WFS-Item object.

Parameters

handle	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

5.13.2.17 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

handle	the pointer to a valid WFS-Item returned by a previous call to get_wfs_catalog_item().
name	the NAME uniquely identifying the required WFS layer.
version	could be "1.0.0" or "1.1.0"; if NULL or invalid "1.1.0" will be assumed.
srid	the preferred SRS to be used for WFS geometries; if negative or mismatching will be simply ignored.
max_features	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().

5.13.2.18 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

handle	the pointer to a valid WFS-Schema returned by a previous call to create_wfs_schema()
index	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

5.13.2.19 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

handle	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

5.13.2.20 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

handle	the pointer to a valid WFS-Column returned by a previous call to get_wfs_schema_column().
name	on completion will contain a pointer to the Column name
type	on completion will contain the datatype set for the Column; could be one of SQLITE_TEXT, SQLITE_INTEGER or SQLITE_FLOAT
nullable	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

5.13.2.21 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

handle	the pointer to a valid WFS-Schema returned by a previous call to create_wfs_schema().
name	on completion will contain a pointer to the GeometryColumn name
type	on completion will contain the GeometryType set for the Column; could be one of GAIA_POINT, GAIA_LINESTRING, GAIA_POLYGON, GAIA_MULTIPOINT, GAIA_MULTILINESTRING, GAIA_MULTIPOLYGON or GAIA_GEOMETRYCOLLECTION
srid	on completion will contain the SRID-value set for the GeometryColumn
dims	on completion will contain the dimensions (2 or 3) set for the GeometryColumn
nullable	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

5.13.2.22 SPATIALITE_DECLARE const char* get_wfs_version (gaiaWFScatalogPtr *handle*)

Return the WFS-Version string as reported by GetCapabilities.

Parameters

handle	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

5.13.2.23 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

sqlite	handle to current DB connection
path_or_url	pointer to some WFS-GetFeature XML Document (could be a pathname or an URL).
alt_describe_uri	an alternative URI for DescribeFeatureType to be used if no one is found within the XML document returned by GetFeature.
layer_name	the name of the WFS layer.
swap_axes	if TRUE the X and Y axes will be swapped

table	the name of the table to be created
pk_column	name of the Primary Key column; if NULL or mismatching then "PK_UID" will be assumed by default.
spatial_index	if TRUE an R∗Tree Spatial Index will be created
rows	on completion will contain the total number of actually imported rows
err_msg	on completion will contain an error message (if any)
progress_↩ callback	pointer to a callback function to be invoked immediately after processing each WFS page (could be NULL)
callback_ptr	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.

5.13.2.24 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

sqlite	handle to current DB connection
path_or_url	pointer to some WFS-GetFeature XML Document (could be a pathname or an URL).
alt_describe_uri	an alternative URI for DescribeFeatureType to be used if no one is found within the XML document returned by GetFeature.
layer_name	the name of the WFS layer.
swap_axes	if TRUE the X and Y axes will be swapped
table	the name of the table to be created
pk_column	name of the Primary Key column; if NULL or mismatching then "PK_UID" will be assumed by default.
spatial_index	if TRUE an R∗Tree Spatial Index will be created
page_size	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).
rows	on completion will contain the total number of actually imported rows
err_msg	on completion will contain an error message (if any)
progress_↩ callback	pointer to a callback function to be invoked immediately after processing each WFS page (could be NULL)

callback_ptr	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

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.

5.13.2.25 SPATIALITE_DECLARE void reset_wfs_http_connection (void)

Resets the libxml2 "nano HTTP": useful when changing the HTTP_PROXY settings.

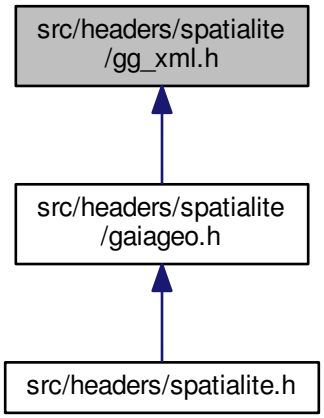
See also

create_wfs_catalog, load_from_wfs, load_from_wfs_paged

5.14 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.

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 gaiaIsValidXmlBlob (const unsigned char *blob, int size)

  Checks if a BLOB actually is a valid XmlBLOB buffer.
- GAIAGEO_DECLARE int gaiaIsCompressedXmlBlob (const unsigned char *blob, int size)

  Checks if a valid XmlBLOB buffer is compressed or not.
- GAIAGEO_DECLARE int gaiaIsIsoMetadataXmlBlob (const unsigned char *blob, int size)

  Checks if a valid XmlBLOB buffer does contain an ISO Metadata or not.
- GAIAGEO_DECLARE int gaiaIsSldSeVectorStyleXmlBlob (const unsigned char *blob, int size)

  Checks if a valid XmlBLOB buffer does contain an SLD/SE Style or not.
- GAIAGEO_DECLARE int gaiaIsSldSeRasterStyleXmlBlob (const unsigned char *blob, int size)

  Checks if a valid XmlBLOB buffer does contain an SLD/SE Style or not.
- GAIAGEO_DECLARE int gaiaIsSldStyleXmlBlob (const unsigned char *blob, int size)

  Checks if a valid XmlBLOB buffer does contain an SLD Style or not.
- GAIAGEO_DECLARE int gaiaIsSvgXmlBlob (const unsigned char *blob, int size)

  Checks if a valid XmlBLOB buffer does contain an SVG Symbol 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 gaiaIsSchemaValidatedXmlBlob (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 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 gaiaIsValidXPathExpression (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.

5.14.1 Detailed Description

Geometry handling functions: XML document.

5.14.2 Function Documentation

5.14.2.1 GAIAGEO_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].

5.14.2.2 GAIAGEO_DECLARE int gaiaIsCompressedXmlBlob (const unsigned char * *blob*, int *size*)

Checks if a valid XmlBLOB buffer is compressed or not.

Parameters

blob	pointer to the XmlBLOB buffer.
size	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, gaiaIsIsoMetadataXmlBlob, gaiaIsSldSeVectorStyleXmlBlob, gaiaIsSldSeRasterStyleXmlBlob, gaiaIsSldStyleXmlBlob, gaiaIsSvgXmlBlob

5.14.2.3 GAIAGEO_DECLARE int gaiaIsIsoMetadataXmlBlob (const unsigned char * *blob*, int *size*)

Checks if a valid XmlBLOB buffer does contain an ISO Metadata or not.

Parameters

blob	pointer to the XmlBLOB buffer.
size	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

5.14.2.4 GAIAGEO_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

blob	pointer to the XmlBLOB buffer.
size	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

gaiaIsValidXmlBlob, gaiaIsSvgXmlBlob, gaiaIsCompressedXmlBlob, gaiaIsIsoMetadataXmlBlob, gaiaIsSldSeVectorStyleXmlBlob, gaiaIsSldSeRasterStyleXmlBlob, gaiaIsSldStyleXmlBlob

5.14.2.5 GAIAGEO_DECLARE int gaiaIsSldSeRasterStyleXmlBlob (const unsigned char * *blob*, int *size*)

Checks if a valid XmlBLOB buffer does contain an SLD/SE Style or not.

Parameters

blob	pointer to the XmlBLOB buffer.
size	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

gaiaIsValidXmlBlob, gaiaIsSchemaValidatedXmlBlob, gaiaIsCompressedXmlBlob, gaiaIsIsoMetadataXmlBlob, gaiaIsSldSeVectorStyleXmlBlob, gaiaIsSldStyleXmlBlob, gaiaIsSvgXmlBlob

5.14.2.6 GAIAGEO_DECLARE int gaiaIsSldSeVectorStyleXmlBlob (const unsigned char * *blob*, int *size*)

Checks if a valid XmlBLOB buffer does contain an SLD/SE Style or not.

Parameters

blob	pointer to the XmlBLOB buffer.
size	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

gaiaIsValidXmlBlob, gaiaIsSchemaValidatedXmlBlob, gaiaIsCompressedXmlBlob, gaiaIsIsoMetadataXml↩Blob, gaiaIsSldSeRasterStyleXmlBlob, gaiaIsSvgXmlBlob

5.14.2.7 GAIAGEO_DECLARE int gaiaIsSldStyleXmlBlob (const unsigned char * *blob*, int *size*)

Checks if a valid XmlBLOB buffer does contain an SLD Style or not.

Parameters

blob	pointer to the XmlBLOB buffer.
size	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

gaiaIsValidXmlBlob, gaiaIsSchemaValidatedXmlBlob, gaiaIsCompressedXmlBlob, gaiaIsIsoMetadataXml↩Blob, gaiaIsSldSeVectorStyleXmlBlob, gaiaIsSldSeRasterXmlBlob, gaiaIsSvgXmlBlob

5.14.2.8 GAIAGEO_DECLARE int gaiaIsSvgXmlBlob (const unsigned char * *blob*, int *size*)

Checks if a valid XmlBLOB buffer does contain an SVG Symbol or not.

Parameters

blob	pointer to the XmlBLOB buffer.
size	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, gaiaIsIsoMetadataXml↩Blob, gaiaIsSldSeVectorStyleXmlBlob, gaiaIsSldStyleXmlBlob, gaiaIsSldSeRasterStyleXmlBlob

5.14.2.9 GAIAGEO_DECLARE int gaiaIsValidXmlBlob (const unsigned char * *blob*, int *size*)

Checks if a BLOB actually is a valid XmlBLOB buffer.

Parameters

blob	pointer to the XmlBLOB buffer.
size	XmlBLOB's size (in bytes).

Returns

TRUE or FALSE

See also

gaiaIsCompressedXmlBlob, gaiaIsSchemaValidatedXmlBlob, gaiaIsIsoMetadataXmlBlob, gaiaIsSldSe↩VectorStyleXmlBlob, gaiaIsSldSeRasterStyleXmlBlob, gaiaIsSldStyleXmlBlob, gaiaIsSvgXmlBlob

5.14.2.10 GAIAGEO_DECLARE int gaiaIsValidXPathExpression (const void * *p_cache*, const char * *xpath_expr*)

Checks if a Text string could be a valid XPathExpression.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
xpath_expr	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

5.14.2.11 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.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
blob	pointer to the input XmlBLOB buffer.
size	input XmlBLOB's size (in bytes).
identifier	the new FileId value to be inserted.
ns_id	prefix corresponding to FileIdentifier NameSpace (may be NULL)
uri_id	URI corresponding to the FileIdentifier NameSpace (may be NULL)
ns_charstr	prefix corresponding to CharacterString NameSpace (may be NULL)
uri_charstr	URI corresponding to CharacterString NameSpace (may be NULL)
new_blob	on completion will contain a pointer to the output XmlBLOB buffer.
new_size	on completion will containg the output XmlBlob's size (in bytes).

Returns

TRUE for success; FALSE for any failure cause.

See also

gaiaIsIsoMetadataXmlBlob, gaiaXmlBlobGetFileId, gaiaXmlBlobSetFileId

Note

the output XmlBLOB corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

5.14.2.12 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

p_cache	a memory pointer returned by spatialite_alloc_connection()
blob	pointer to the inputXmlBLOB buffer.
size	input XmlBLOB's size (in bytes).
identifier	the new ParentId value to be inserted.
ns_id	prefix corresponding to FileIdentifier NameSpace (may be NULL)
uri_id	URI corresponding to the FileIdentifier NameSpace (may be NULL)
ns_charstr	prefix corresponding to CharacterString NameSpace (may be NULL)
uri_charstr	URI corresponding to CharacterString NameSpace (may be NULL)
new_blob	on completion will contain a pointer to the output XmlBLOB buffer.
new_size	on completion will containg the output XmlBlob's size (in bytes).

Returns

TRUE for success; FALSE for any failure cause.

See also

gaiaIsIsoMetadataXmlBlob, gaiaXmlBlobGetParentId, gaiaXmlBlobSetParentId

Note

the returned XmlBLOB corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

5.14.2.13 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.

Parameters

blob	pointer to the input XmlBLOB buffer.
in_size	input XmlBLOB's size (in bytes).
compressed	if TRUE the returned XmlBLOB will be zip-compressed.

result	on completion will containt a pointer to the output XmlBLOB: NULL on failure.
out_size	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].

5.14.2.14 GAIAGEO_DECLARE char* gaiaXmlBlobGetAbstract (const unsigned char * *blob*, int *size*)

Return the Abstract from a valid XmlBLOB buffer.

Parameters

blob	pointer to the XmlBLOB buffer.
size	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, gaiaIsSld↩StyleXmlBlob

Note

the returned Abstract corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

5.14.2.15 GAIAGEO_DECLARE int gaiaXmlBlobGetDocumentSize (const unsigned char * *blob*, int *size*)

Return the XMLDocument size (in bytes) from a valid XmlBLOB buffer.

Parameters

blob	pointer to the XmlBLOB buffer.
size	XmlBLOB's size (in bytes).

Returns

the XMLDocument size (in bytes) for any valid XmlBLOB; -1 if the BLOB isn't a valid XmlBLOB.

5.14.2.16 GAIAGEO_DECLARE char* gaiaXmlBlobGetEncoding (const unsigned char * *blob*, int *size*)

Return the Charset Encoding from a valid XmlBLOB buffer.

Parameters

blob	pointer to the XmlBLOB buffer.
size	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.

5.14.2.17 GAIAGEO_DECLARE char* gaiaXmlBlobGetFileId (const unsigned char * *blob*, int *size*)

Return the FileIdentifier from a valid XmlBLOB buffer.

Parameters

blob	pointer to the XmlBLOB buffer.
size	XmlBLOB's size (in bytes).

Returns

the FileIdentifier for any valid XmlBLOB containing a FileIdentifier; NULL in any other case.

See also

gaiaIsIsoMetadataXmlBlob, gaiaXmlBlobSetFileId, gaiaXmlBlobAddFileId

Note

the returned FileIdentifier corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

5.14.2.18 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

blob	pointer to the XmlBLOB buffer.
size	XmlBLOB's size (in bytes).
blob_geom	on completion this variable will contain a pointer to the returned Geometry Buffer (NULL if no Geometry was defined within the XmlBLOB)
blob_size	on completion this variable will contain the size (in bytes) of the returned Geometry Buffer

See also

gaiaIsIsoMetadataXmlBlob

Note

the returned Geometry Buffer corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

5.14.2.19 GAIAGEO_DECLARE char* gaiaXmlBlobGetLastParseError (const void * *p_cache*)

Return the most recent XML Parse error/warning (if any)

Parameters

ptr	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, gaiaIsValidXPathExpression, gaiaXmlBlobGetLastXPathError

Note

the returned error/warning message corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

5.14.2.20 GAIAGEO_DECLARE char* gaiaXmlBlobGetLastValidateError (const void * *p_cache*)

Return the most recent XML Validate error/warning (if any)

Parameters

p_cache	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, gaiaIsValidXPathExpression, gaiaXmlBlobGetLastXPathError

Note

the returned error/warning message corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

5.14.2.21 GAIAGEO_DECLARE char* gaiaXmlBlobGetLastXPathError (const void * *p_cache*)

Return the most recent XPath error/warning (if any)

Parameters

p_cache	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

gaiaXmlBlobGetLastParseError, gaiaXmlBlobGetLastValidateError, gaiaIsValidXPathExpression

Note

the returned error/warning message corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

5.14.2.22 GAIAGEO_DECLARE char* gaiaXmlBlobGetName (const unsigned char * *blob*, int *size*)

Return the Name from a valid XmlBLOB buffer.

Parameters

blob	pointer to the XmlBLOB buffer.
size	XmlBLOB's size (in bytes).

Returns

the Name for any valid XmlBLOB containing a Name; NULL in any other case.

See also

gaiaIsIsoMetadataXmlBlob, gaiaIsSldSeVectorStyleXmlBlob, gaiaIsSldSeRasterStyleXmlBlob, gaiaIsSld↩StyleXmlBlob

Note

the returned Name corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

5.14.2.23 GAIAGEO_DECLARE char* gaiaXmlBlobGetParentId (const unsigned char * *blob*, int *size*)

Return the ParentIdentifier from a valid XmlBLOB buffer.

Parameters

blob	pointer to the XmlBLOB buffer.
size	XmlBLOB's size (in bytes).

Returns

the ParentIdentifier for any valid XmlBLOB containing a ParentIdentifier; NULL in any other case.

See also

gaiaIsIsoMetadataXmlBlob, gaiaXmlBlobSetParentId, gaiaXmlBlobAddParentId

Note

the returned ParentIdentifier corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

5.14.2.24 GAIAGEO_DECLARE char* gaiaXmlBlobGetSchemaURI (const unsigned char * *blob*, int *size*)

Return the SchemaURI from a valid XmlBLOB buffer.

Parameters

blob	pointer to the XmlBLOB buffer.
size	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.

5.14.2.25 GAIAGEO_DECLARE char* gaiaXmlBlobGetTitle (const unsigned char * *blob*, int *size*)

Return the Title from a valid XmlBLOB buffer.

Parameters

blob	pointer to the XmlBLOB buffer.
size	XmlBLOB's size (in bytes).

Returns

the Title for any valid XmlBLOB containing a Title; NULL in any other case.

See also

gaiaIsIsoMetadataXmlBlob, gaiaIsSldSeVectorStyleXmlBlob, gaiaIsSldSeRasterStyleXmlBlob, gaiaIsSldStyleXmlBlob

Note

the returned Title corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

5.14.2.26 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.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
blob	pointer to the input XmlBLOB buffer.
size	input XmlBLOB's size (in bytes).
identifier	the new FileId value to be set.
new_blob	on completion will contain a pointer to the output XmlBLOB buffer.
new_size	on completion will containg the output XmlBlob's size (in bytes).

Returns

TRUE for success; FALSE for any failure cause.

See also

gaiaIsIsoMetadataXmlBlob, gaiaXmlBlobGetFileId, gaiaXmlBlobAddFileId

Note

the output XmlBLOB corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

5.14.2.27 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.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
blob	pointer to the inputXmlBLOB buffer.
size	input XmlBLOB's size (in bytes).
identifier	the new ParentId value to be set.
new_blob	on completion will contain a pointer to the output XmlBLOB buffer.
new_size	on completion will containg the output XmlBlob's size (in bytes).

Returns

TRUE for success; FALSE for any failure cause.

See also

gaiaIsIsoMetadataXmlBlob, gaiaXmlBlobGetParentId, gaiaXmlBlobAddParentId

Note

the returned XmlBLOB corresponds to dynamically allocated memory: so you are responsible to free() it before or after.

5.14.2.28 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.

Parameters

blob	pointer to the XmlBLOB buffer.
size	XmlBLOB's size (in bytes).
indent	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 intenting (max. 8); ZERO means that the whole XML Document will consist of a single line.
result	pointer to the memory buffer containing the XML Document
res_size	dimension (in bytes) of the XML Document memory buffer (both values will be passed back after successful completion).

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.

5.14.2.29 GAIAGEO_DECLARE char* gaiaXmlGetInternalSchemaURI (const void * *p_cache*, const unsigned char * *xml*, int *xml_len*)

Return the Internal SchemaURI from a valid XmlDocument.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
xml	pointer to the XML document
xml_len	lenght 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.

5.14.2.30 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.

Parameters

path_or_url	pointer to the external XML Document (could be a pathname or an URL).
result	on completion will containt a pointer to a BLOB: NULL on failure.
size	on completion this variable will contain the BLOB's size (in bytes).
parsing_errors	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].

5.14.2.31 GAIAGEO_DECLARE int gaiaXmlStore (const unsigned char * *blob*, int *size*, const char * *path*, int *indent*)

Stores an external XML Document.

Parameters

blob	pointer to the XmlBLOB buffer.
size	XmlBLOB's size (in bytes).
path	pathname of the export file

indent	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 intenting (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].

5.14.2.32 GAIAGEO_DECLARE char* gaiaXmlTextFromBlob (const unsigned char * *blob*, int *size*, int *indent*)

Extract an XMLDocument from within an XmlBLOB buffer.

Parameters

blob	pointer to the XmlBLOB buffer.
size	XmlBLOB's size (in bytes).
indent	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 intenting (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.

5.14.2.33 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.

Parameters

p_cache	a memory pointer returned by spatialite_alloc_connection()
xml	pointer to the XML document (XmlBLOB payload).
xml_len	lenght of the XML document (in bytes).
compressed	if TRUE the returned XmlBLOB will be zip-compressed.
schemaURI	if not NULL the XML document will be assumed to be valid only if it successfully passes a formal Schema valitadion.
result	on completion will containt a pointer to XmlBLOB: NULL on failure.
size	on completion this variable will contain the XmlBLOB's size (in bytes)
parsing_errors	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.
schema_validation_errors	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 6

Example Documentation

6.1 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

/*

demo1.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 "config.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 resul 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 =
                                    gaiaFromSpatiaLiteBlobWkb (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_MULTIPOINT)
```

```
                                            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)
                                          {
#ifndef OMIT_GEOS               /* GEOS is required */
                                            gaiaGeomCollLength (geom,
                                                                &measure);
                                            printf (" length=%1.2f",
                                                    measure);
#else
                                            printf
                                                (" length=?? [no GEOS support available]");
#endif /* GEOS enabled/disabled */
                                          }
                                        if (geom_type == GAIA_POLYGON ||
                                            geom_type ==
                                            GAIA_MULTIPOLYGON)
                                          {
#ifndef 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;
}
```

6.2 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,

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), P
```

```
/*

demo2.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 "config.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_MULTIPOINT)
        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\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\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);


#ifndef OMIT_GEOS               /* GEOS must be enabled */

/*
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\t\tDimension=%d IsValid=%d\n",
            geom_type (gaiaGeometryType (geo_pt)),
            gaiaDimension (geo_pt), gaiaIsValid (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 (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 (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 (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 (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 (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 (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;
}
```

6.3 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
        ("\nnow 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;
}
```

6.4 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
        ("\nnow 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;
}
```

6.5 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_MULTIPOINT:
                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->AttributeFieldName);
                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)
#if defined(_WIN32) || defined(__MINGW32__)
/* CAVEAT: M$ runtime doesn't supports %lld for 64 bits */
                    printf ("\t\t\tIntRange %I64d / %I64d\n",
#else
                    printf ("\t\t\tIntRange %lld / %lld\n",
#endif
                            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\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;

    if (argc > 1 || argv[0] == NULL)
        argc = 1;               /* silencing stupid compiler warnings */

    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