# Open Geospatial Consortium Inc.

Date: 2013-08-09

Reference number of this document: 13-073

Version: 1.0.0

Category: OpenGIS® RFC Comments

Editor: Paul Daisey

# OGC 12-128r1 Version 0.2.0 GeoPackage Implementation Specification

# RFC Comments & Responses



**Warning**

This document is not an OGC Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

| | |
|---|---|
| Document type: | **OGC**© RFC Comments |
| Document subtype: | |
| Document stage: | Draft |
| Document language: | English |

# Contents

Page

# Foreword

The initial informal "OGC 12-128r1 GeoPackage Candidate Implementation Specification" Request For Comment (RFC) period ran from the 8th of January 2013 to 8th of February 2013. This document captures all the comments received and the subsequent responses from the Standards Working Group (SWG).

*Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium Inc. shall not be held responsible for identifying any or all such patent rights.*

*Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.*

# OGC 12-128r1Version 0.2.0 GeoPackage Candidate Standard: Received Comments

Below is the list of all comments received during the RFC (8th January – 8th February 2013).

## 1 Comment: Ragi Burham 2013-01-10

**PART A**

1. Evaluator:

Ragi Burhum
CEO AmigoCloud Inc
ragi@burhum.com
http://www.burhum.com

2. Submission:

OGC 12-128r1 Version: 0.2.0 - OpenGIS® GeoPackage Implementation Specification

**PART B**

### 1.1 Comment: lib spatialite + GEOS

1. Requirement: General - Using Spatialite as a reference implementation
2. Implementation Specification Section number: General
3. Criticality: major
4. Comments/justifications for changes:

Using libspatialite + GEOS + proj as a reference implementation.

It is truly sad that libspatialite + GEOS was chosen as a reference implementation since, because of licensing issues, it cannot be used in *any* iOS devices (iPhone, iPad, etc).

For background, please see http://blog.burhum.com/post/38236943467/your-lgpl-license-is-completely-destroying-ios-adoption

Choosing sqlite by itself - a project with an Open Source project with a more liberal license that allows usage in *any* phone/tablet would have been a far better choice.

#### 1.1.1 SWG Response

Thank you for your comment. The GeometryEncodingFormatSelection discussion item was resolved by selection of a SQL BLOB binary geometry format other than that used by SpatiaLite, so the GeoPackage spec is now based solely on SQLite and no longer depends upon or uses SpatiaLite as a reference implementation. So the GEOS lib may be used by some implementation of a GeoPackage, but is no longer specified, and completely optional.

## 2 Comments: Evan Rouault 2013-01-12

**Part A**

1. Evaluator:
   Even Rouault, even.rouault at mines-paris dot org
   GDAL/OGR contributor
2. Submission:
OGC 12-128r1 Version: 0.2.0 - OpenGIS® GeoPackage Implementation Specification

**PART B**

### 2.1 Comment: Two parts to specification

1. Requirement: General
2. Implementation Specification Section number: General
3. Criticality: Minor
4. Comments/justifications for changes:

It could perhaps be made more clear in the preface/introduction/scope that the specification contain 2 different things :

- storage requirements that make possible the exchange of data
- and various SQL operations that can be used on the data stored in a geopackage, but are optionnal to just read or write the data.

The Spatialite reference implementation, or any other implementation with equivalent SQL functions, is in no way necesserary to be able to use the vector information in a GeoPackage. What you need is to support the Spatialite geometry blob format. This is for example done in the GDAL/OGR library.

Linked to that, I'm uneasy with the term "container" as it is used in the document. It makes sense in REQ 1 to define it as a file, but later requirements mention properties that cannot be supported by a container in itself, but rather by a software implementation that provides capabilities. For example REQ 4 "The GeoPackage container shall provide a “C” language CLI": a .geopackage file doesn't provide a C language CLI, it is libsqlite3 that provides that. Perhaps different terms could be used to separate the format specification, and the services that a software implementation handling .geopackage should offer.

#### 2.1.1 SWG Response

Thank you for your comment. The Introduction and Terms and Definitions clauses have been rewritten to identify separate GeoPackage **file** and GeoPackage **SQLite Extension** (code) components of a GeoPackage.

### 2.2 Comment: Table Diagram max_y column data type

1. Requirement:
2. Implementation Specification Section number: 7 Table Diagram
3. Criticality: Editorial
4. Comments/justifications for changes:

The max_y field of table geopackage_contents should be of type REAL, not BLOB

#### 2.2.1 SWG Response

Thank you for your comment. Agreed. The table diagram, which is now in the Introduction, has been corrected. The table name is now gpkg_contents.

### 2.3 Comment: SQLite version 3.7

1. Requirement: REQ 9
2. Implementation Specification Section number: "6.2 - Reference Implementation"
3. Criticality: Minor
4. Comments/justifications for changes:

Is SQLite 3.7 really necessary to define the storage file format ?

According to http://www.sqlite.org/formatchng.html , the last file format change was in SQLite 3.6.0. Many enterprise Linux distributions ship with SQLite 3.6.X (e.g. SQLite 3.6.20 on RHEL 6).

#### 2.3.1 SWG Response

Thank you for your comment. Resolution of the DatabaseFormat discussion item included the following statement: "The GeoPackage file format shall be an SQLite [9] database file with the version 3 file format [10][11]" which is now the beginning of Req 1.

### 2.4 Comment: SQLite file header

1. Requirement: REQ 10
2. Implementation Specification Section number: 6.2 - Reference Implementation
3. Criticality: Minor
4. Comments/justifications for changes:

What's the usefulness of redefining the header of a SQLite3 database ? This is implied by REQ 9, and there are so many other requirements that should be written to specify the interoperability with a SQLite3 database, if you don't use the "SQLite3 reference implementation". This might be just an informational note however, to help writing a preliminary check that the file is a good candidate for being a geopackage container.

#### 2.4.1 SWG Response

Thank you for your comment. There is no GeoPackage redefinition of the header in an SQLite database. http://www.sqlite.org/fileformat2.html specifies that the header begins with "SQLite format 3". The previous REQ 9 and REQ 10 have been dropped by resolution of the DatabaseFormat discussion item and replaced by a new Req 1: "The GeoPackage file format SHALL be a SQLite [9] database version 3 format [10][11] file, the first 16 bytes of which contain “SQLite format 3”1 in ASCII.".

### 2.5 Comment: geopackage_contents table default values

1. Requirement:
2. Implementation Specification Section number: 8.2 - Geopackage Contents Table
3. Criticality: Minor
4. Comments/justifications for changes:

There are inconsistencies between the default values in "Table – Geopackage Contents Table or View Definition" and "Table 2 – geopackage_contents Table Definition SQL" for columns "identifier" and "description". I haven't verified, but I doubt that a default value makes sense for identifier if it is supposed to be UNIQUE. And for description, table 1 mentions "", whereas table 2 mentions "none".

#### 2.5.1 SWG Response

Thank you for your comment. Agreed. The identifier column no longer has a default value. The default value for the description column is now “” in both tables. Subsequent draft revisions (now 12-128r8) have been reorganized, with renumbering of requirements, clauses and tables since this resolution was approved. The relevant tables are now table 2 and table 22, and the database table has been renamed gpkg_contents.

### 2.6 Comment: “gb” storage type

1. Requirement:
2. Implementation Specification Section number: 9.4 - Feature Tables
3. Criticality: Major
4. Comments/justifications for changes:

This paragraph can lead the implementator of another implementation in error, since you get the false impression that the "gb" storage type is WKB, whereas it is in fact later explained (page 46) to be the spatialite blob format. I think this should be more clearly stated right here.

#### 2.6.1 SWG Response

Thank you for your comment. The draft GeoPackage spec has been revised to remove references to the SF/SQL gB and gS storage architectures.

### 2.7 Comment: Spatial Index Rtree Implementation

1. Requirement: REQ 25

2. Implementation Specification Section number: 9.4 - Feature Tables
3. Criticality: Major
4. Comments/justifications for changes:

It is unfortunate that spatial index is mentionned, but not specified !
Different implementations of GeoPackage could have incompatible spatial index.
Why not just specifying the implementation done in Spatialite ?, i.e. based on
the SQLite3 R-Tree extension.

If that was done, it should be mentionned in REQ 9 that the SQLite3
implementation to use must have R-Tree support, which is an optional
configuration to enable at build time.

#### 2.7.1 SWG Response

Thank you for your comment. GeoPackage spatial index implementation based on SQLite rtrees is now specified in clause 3.1.3.

### 2.8 Comment: Spatial index performance requirements

1. Requirement: REQ 26 and REQ 27
2. Implementation Specification Section number: 9.4 - Feature Tables
3. Criticality: Minor
4. Comments/justifications for changes:

I fail to see why the various quantities (1 degree bounding box, 1000
features, etc...) are mentionned. REQ 27 could be just written "Spatial
queries in a GeoPackage shall provide the same results with and without a
spatial index." And REQ 28 should be just removed.

You can certainly find artificial cases of data where the spatial index will
lead to slower request times.

#### 2.8.1 SWG Response

Thank you for your comment. These requirements were dropped during subsequent reorganization of the draft specification and no longer appear in 12-128r8.

### 2.9 Comment: Spatialite V3 hacks

1. Requirement: REQ 44
2. Implementation Specification Section number: 9.6 - Reference Implementation
3. Criticality: Minor
4. Comments/justifications for changes:

Why not just specifying Spatialite V4 as the minimum version ? The various
hacks given at page 46 to turn a Spatialite V3 into a pseudo V4 are just ugly
and lead to confusion. They could be mentionned on non-official documents, such
as the Spatialite wiki, but it is awkward to see them mentionned in a formal
document, like that one.

#### 2.9.1 SWG Response

Thank you for your comment. Those hacks predated Spatialite v4. They have now been removed. Note that subsequent draft revisions (now 12-128r8) have been reorganized, with renumbering of requirements, clauses and tables.

### 2.10 Comment: SpatiaLite compressed geometries

1. Requirement: REQ 47
2. Implementation Specification Section number: 9.6 - Reference Implementation
3. Criticality: Minor
4. Comments/justifications for changes:

It would be worth mentionning that the "Compressed geometries" described in http://www.gaia-gis.it/gaia-sins/BLOB-Geometry.html are not valid for a baseline GeoPackage.

#### 2.10.1 SWG Response

Thank you for your comment. The GeometryEncodingFormatSelection discussion item was resolved by selection of a SQL BLOB binary geometry format other than that used by SpatiaLite, so the GeoPackage spec is now based solely on SQLite, no longer depends upon or uses SpatiaLite as a reference implementation, and no longer offers a "compressed geometries" option.

### 2.11 Comment: compr_qual_factor and georectification columns

1. Requirement:
2. Implementation Specification Section number: 10.2 Raster Columns
3. Criticality: Minor
4. Comments/justifications for changes:

I'd suggest making the compr_qual_factor and georectification columns nullable, or have another default value meaning "unknown". If you convert an existing set of tiled JPEGs into a GeoPackage, you generally don't know the quality associated (it is certainly different of 100 %, but is it 25% or 75% ??). Same for georectification. Those columns provide non essential information.

#### 2.11.1 SWG Response

Thank you for your comment. These columns and the table that contained them have been dropped from the draft GeoPackage specification.

### 2.12 Comment: Raster metadata table name

1. Requirement:
2. Implementation Specification Section number: 10.7 Rasters or Tiles Table Metadata
3. Implementation Criticality: Minor
4. Comments/justifications for changes:

Actually, in RasterLite (current state of public sources at least), the

raster metadata table is suffixed by _metadata, not _rt_metadata.

#### 2.12.1 SWG Response

Thank you for your comment. Agree that the suffix is not the same, the convention referenced was the use of a table name as the prefix. The NOTE with this description has been removed. Subsequent draft revisions (now 12-128r8) have been reorganized, with renumbering of requirements, clauses and tables.

### 2.13 Comment: gpkgBboxToTiles() implementation

1. Requirement:
2. Implementation Specification Section number: 10.8.10 gpkgBboxToTiles SQL Function
3. Criticality: Major
4. Comments/justifications for changes:

I'm not sure how gpkgBboxToTiles() can be implemented as a SQLite function. To the best of my knowledge, and according to http://www.sqlite.org/c3ref/create_function.html, custom SQLite functions can return a single value (string, integer, real, text, blob), potentially operating as aggregates on several source rows, but not generating several rows. What is the "set of integers" return value mentionned here : a string of ids, separated by commas or spaces ?

#### 2.13.1 SWG Response

Thank you for your comment. This function has been dropped from the specification.

### 2.14 Comment: Tile matrix set bounds

1. Requirement:
2. Implementation Specification Section number:
3. Implementation Criticality: Major
4. Comments/justifications for changes:

In the context of a raster pyramid whose more resoluted level is e.g. a world region, but that has pyramids up to the "whole world" zoom level, my understanding is that the bounding box recorded into geopackage_contents would be -180,-90,180,90. Not particularly useful. It would be good to have a direct way of knowing the extent of the more resoluted data, without having to run a query on the _rt_metadata table, like "SELECT MIN(min_x), MIN(min_y), MAX(max_x), MAX(max_y) FROM xxx_rt_metadata", which can be slow. Furthermore, if the edge tiles have padding, you would get that padding in the returned extent, which might be undesirable.

In the tile_matrix_metadata table, this could perhaps be implemented by adding min_x, min_y, max_x, max_y columns, which would represent the extent of the significant area (that is to say without any potential padding).

#### 2.14.1 SWG Response

Thank you for your comment. The GeoPackage SWG has decided not to add any image metadata to GeoPackage tables. Applications should rely on image metadata in an image format that contains it, such as NITF, or put required image metadata in the metadata table and relate it to raster or tile images with metadata_reference records.

### 2.15 Comment: Proposed band_count column

1. Requirement:
2. Implementation Specification Section number: 10.2 Raster Columns
3. Implementation Criticality: Minor
4. Comments/justifications for changes:

In the raster_columns table, it might be appropriate to have a
"band_count" column, that would indicate the maximal number of raster bands
(GDAL terminology, or also called image channels) among all the tiles /
rasters associated with the table with the raster column.

For example, 1 for grey-level raster, 3 for RGB images, 4 for RGBA
images. This would be usefull for example when converting a GeoPackage into a
old-fashioned GIS format, like GeoTIFF (PNG8 without transparency would be
considered as 3 bands after color table expansion, and with transparency as 4
bands).

#### 2.15.1 SWG Response

Thank you for your comment. The GeoPackage SWG has decided to remove raster metadata columns such as compr_qual_factor and georectification from the SQL table model. Applications for use cases that require such metadata should use metadata in an imagery format that includes it with the image. All GeoPackage image formats include band information.

### 2.16 Comment: Manifest column default value

1. Requirement:
2. Implementation Specification Section number: 13.2 Manifest Content Model and Table Definition
3. Implementation Criticality: Minor
4. Comments/justifications for changes:

Is the default value proposed for the manifest column (empty string)
considered as a valid value that will comply with REQ 78 (I think not) ?

#### 2.16.1 SWG Response

Thank you for your comment. The manifest table has been removed from the GeoPackage specification. Metadata such as OGC Context documents stored in the metadata table may be used to provide links to online sources of GeoPackage data.

# 3 Comments: Darrell Fuhriman 2013-01-15

**Part A**

1. Evaluator:

Darrell Fuhriman, darrell@garnix.org

2. Submission:

OGC 12-128r1 Version: 0.2.0 - OpenGIS® GeoPackage Implementation Specification

**Part B**

## 3.1 Comment: Bounding box units of measure

1. Requirement: Core, #13
2. Implementation Specification Section number: 8.2
3. Implementation Criticality: Minor
4. Comments/justifications for changes:

The geopackage_contents table includes four columns used for the bounding box, but does not specify the units to be used, only the defaults. The spec should be updated to specifically list the units.

Points for clarification:
1) are the values to always be in lat/long?
1a) If so, what datum? (Yes, I realized these values are not meant to be exact, but it doesn't hurt be clear)
2) are the values to be in the SRID specified in the same row?

If the values are to be specified always in lat/long, then recommend renaming columns to reflect this:
min_x -> min_long
max_x -> max_long
min_y -> min_lat
max_y -> max_lat

### 3.1.1 SWG Response

Thank you for your comment. The intent was that the min/max x/y values would be in the distance units of a projected coordinate reference system specified by the srid, or decimal degrees if the srid specified a geographic crs. Resolution: In clause 8.2 add a new last sentence to the first paragraph on page 16, and an informative NOTE following it. Rename the following NOTE to NOTE2.

"If the srid column value references a geographic coordinate reference system (CRS), then the min/max x/y values are in decimal degrees; otherwise, the srid references a projected CRS and the min/max x/y values are in the units specified by that CRS."

"NOTE1: the "0" default value for the srid column is for an undefined geographic CRS. See clause 9.2 below." Subsequent draft revisions (now 12-128r8) have been reorganized, with renumbering of requirements, clauses and tables since this resolution was approved.

### 3.2 Comment: Include feature styling information

1. Requirement: NA
2. Implementation Specification Section number: NA
3. Implementation Criticality: Major
4. Comments/justifications for changes:

I believe there should be an optional extension that allows for the inclusion of user-defined styling information.

While, I do not believe that the Geopackage format should mandate a specific styling language, such as SLD or CartoCSS, I do think there should be a standard location where an application can look for such data.

The implementation should be highly generic, the intention is to allow standards to evolve and gain experience before codifying anything, but still having a single "go to table" for finding styling suggestions.

As an intro suggestion, I imagine there being a table, say "geopackage_styles" consisting of four columns:

table_name  text NOT NULL
style_name  text NOT NULL
style_version text
style_data text NOT NULL

The primary key would be a composite key consisting of "table_name, style_name", thus allowing for one table to have multiple styles associated with in different styling languages.

For example, a table might contain:

| my_table | cartoCSS | NULL  | "some carto css"  |
| my_table | SLD  | 1.1.0 | "<BIG string of XML/>"|

Any styling information included must apply to the named table/view, and only that table/view. Including styling suggestions for a table/view is purely optional, and the "geopackage_styles" table need not exist at all.

#### 3.2.1 SWG Response

Thank you for your comment.  Adding styling information to a GeoPackage has been added as a potential future work item.

### 3.3 Comment: Use of SpatiaLite + GEOS

1. Requirement: NA
2. Implementation Specification Section number: 9.6
3. Implementation Criticality: Major

4. Comments/justifications for changes:
5.

I am concerned that the reference implementation SpatialLite, essentially mandates the inclusion of the GEOS library. While I have nothing against the GEOS library, quite the opposite, this presents real problems for people who must statically link the libraries, something that is expressly forbidden by the GEOS Library License (LGPL).

For reference, see: http://blog.burhum.com/post/38236943467/your-lgpl-license-is-completely-destroying-ios-adoption

I strongly encourage the OGC to create a minimal reference implementation, contains *only* the core requirements, and makes use only of libraries which can be statically linked.

#### 3.3.1 SWG Response

Thank you for your comment. The GeometryEncodingFormatSelection discussion item was resolved by selection of a SQL BLOB binary geometry format other than that used by SpatiaLite so the GeoPackage spec is now based solely on SQLite and no longer depends upon or uses SpatiaLite as a reference implementation. So the GEOS lib may be used by some implementation of a GeoPackage, but is completely optional.

## 4 Comments: Evan Rouault 2013-01-22.

**Part A**

1. Evaluator:
   Even Rouault, even.rouault at mines-paris dot org
   GDAL/OGR contributor

2. Submission:

OGC 12-128r1 Version: 0.2.0 - OpenGIS® GeoPackage Implementation Specification

**Part B**

### 4.1 Comment: REQ 71 incorrect clause reference

1. Requirement: REQ 71
2. Implementation Specification Section number: 11.2 XML Metadata Table
3. Implementation Criticality: Editorial
4. Comments/justifications for changes:

REQ 71 should refer to clause 11.2 and not 11.1

#### 4.1.1 SWG Response

Thank you for your comment. Agreed. Change made in r2. But this change is no longer apparent in 12-128r8 due to rewording and renumbering of requirements in subsequent draft revisions.

### 4.2 Comment: REQ 74 incorrect clause reference

1. Requirement: REQ 74
2. Implementation Specification Section number: 11.3 Metadata Reference Table
3. Implementation Criticality: Editorial
4. Comments/justifications for changes:

REQ 74 should refer to clause 11.3 and not 11.2

#### 4.2.1 SWG Response

Thank you for your comment. Agreed, also applies to REQ 75. Changes made to r2. But this change is no longer apparent in 12-128r8 due to rewording and renumbering of requirements in subsequent draft revisions.

### 4.3 Comment: Metadata Reference Table Specified Values

1. Requirement: REQ 74
2. Implementation Specification Section number: 11.3 Metadata Reference Table
3. Implementation Criticality: Major
4. Comments/justifications for changes:

What are the "values as specified in clause 11.2" (11.3 likely, see previous comment) mentionned in REQ 74 ? §11.3 only defines the table and triggers, and some constraints (REQ 76). Are there compulsory rows to insert in the metatable_reference table ? My understand is that an empty metatable_reference table is valid.

#### 4.3.1 SWG Response

Thank you for your comment. The "values as specified" clause has been removed from the requirement, and the following statement added: “The metadata_reference table is not required to contain any rows." Subsequent draft revisions (now 12-128r8) have been reorganized, with renumbering of requirements, clauses and tables since this resolution was approved.

### 4.4 Comment: Manifest Table Name

1. Requirement: REQ 77
2. Implementation Specification Section number: 13.2 Manifest XML Schema
3. Implementation Criticality: Major
4. Comments/justifications for changes:

There is an inconsistency between the title of Table 56 (*ows_manifest* table) and Table 57 (*ows_manifest* Table Definition SQL), and the rest of the document where the name of the table is just *manifest*

#### 4.4.1 SWG Response

Thank you for your comment. The manifest table was dropped during subsequent revision of the draft specification and no longer appears in 12-128r8.

### 4.5 Comment: Manifest table records

1. Requirement: REQ 78
2. Implementation Specification Section number: 13.3 Manifest XML Schema
3. Implementation Criticality: Major
4. Comments/justifications for changes:

Do you confirm that having a record in the manifest table is *not* required (and that only creating the table structure is actually required) to create a valid GeoPackage ? It is my understanding of the "Extension" status of REQ 78, but this has generated some discussion on spatialite-users list recently. Inserting a record with a valid XML content might be difficult in a automated conversion process.

#### 4.5.1 SWG Response

Thank you for your comment. The manifest table has been removed from the GeoPackage specification. Metadata such as OGC Context documents stored in the metadata table may be used to provide links to online sources of GeoPackage data. The metadata table metadata column is specified as not null, with mime types other than XML are now supported by a metadata table mime type column.

## 5 Comment: Norman Barker 2013-02-04

**Part A**

1. Evaluator:
   Norman Barker <norman@cloudant.com>

2. Submission:

OGC 12-128r1 Version: 0.2.0 - OpenGIS® GeoPackage Implementation Specification

**Part B**

### 5.1 Comment: Propose streaming protocol updates

1. Requirement: Core, #2
2. Implementation Specification Section number: 6.1 Capabilities
3. Implementation Criticality: Major
4. Comments/justifications for changes

This format is specified as a SQLite database which will require a import for all other databases (IndexedDB, TouchDB etc). A mobile device may not have sufficient resources (memory and storage) to transcode that file format on the device and the file will either

have to be converted before download or (slower) adapters to read the format for other databases will be required.

An additional requirement that Cloudant recommends to assist with this issue is;

Req 2i. The Geopackage shall in the future support incremental updates using a streaming protocol (synchronisation). For example this protocol can be JSON over HTTP(S) for features or a cache for web tiles.

#### 5.1.1 SWG Response

Thank you for your suggestion. It is not in scope for the current version of the GeoPackage spec, but may be in scope for a subsequent version, and has been added as a potential future work item.

### 5.2 Comment: Remove Requirement #3

1. Requirement: Core, #3
2. Implementation Specification Section number: 6.1 Capabilities
3. Implementation Criticality: Major
4. Comments/justifications for changes

In a disconnected environment a model of eventual consistency with multi-version concurrency control is another valid approach. Cloudant ask that this requirement be removed as it is potentially confusing when synchronisation is supported in the specification

#### 5.2.1 SWG Response

Thank you for your comment. Req #3 has been removed by changes to resolve the DatabaseFormat discussion item by making SQLite the file format for GeoPackages instead of the GeoPackage container reference implementation..

### 5.3 Comment: Support Common Query Language

1. Requirement: General
2. Implementation Specification Section number: 6.1, p. 37
3. Implementation Criticality: Minor
4. Comments/justifications for changes

GeoPackage container conformance with current ISO/IEC 9075 (SQL) standards [3][4][5][6][7] would be optimal, but at a minimum the GeoPackage container shall support SQL-92 [2], this seems to deviate from other OGC specifications that recommend clients and servers support Common Query Language (CQL). There is overlap with SQL, but CQL is a defined BNF grammar within existing OGC specifications and it would provider greater interoperability if this specification followed existing OGC standards.

#### 5.3.1 SWG Response

Thank you for your comment. CQL is recommended for OGC Web Service clients and servers. However, GeoPackage is a data container and associated SQL API, not a web service. The existing OGC SF/SQL standards [11][12][13] on which GeoPackage support for vector features is based are extensions of the SQL standards you mention. So although the GeoPackage specification does not specify CQL, it does follow other OGC standards.

### 5.4 Comment: FOR EACH ROW triggers

1. Requirement: Core, #7
2. Implementation Specification Section number: 6.1 Capabilities
3. Implementation Criticality: Major
4. Comments/justifications for changes

This is an implementation requirement and should not be required.

#### 5.4.1 SWG Response

Thank you for your comment. Req #7 has been removed by changes to resolve the DatabaseFormat discussion item by making SQLite the file format for GeoPackages instead of the GeoPackage container reference implementation.

### 5.5 Comment: Streaming synchronisation protocol

1. Requirement: Core, #10
2. Implementation Specification Section number: 6.2 Reference implementation
3. Implementation Criticality: Minor
4. Comments/justifications for changes

See recommendation Req 2i above that an additional streaming synchronisation protocol be defined.

#### 5.5.1 SWG Response

Thank you for your suggestion. The SWG does not consider streaming synchronization to be in scope for the current version of the GeoPackage specification. It has been added as a potential future work item for a future version of the spec.

### 5.6 Comment: UML Table Diagram

1. Requirement: General
2. Implementation Specification Section number: 7. Table Diagram
3. Implementation Criticality: Minor
4. Comments/justifications for changes

Recommend that this diagram be a UML diagram so that a GeoJSON model can be defined in addition to a relational SQLite model.

#### 5.6.1 SWG Response

Thank you for your comment. Thank you for your comment. The table diagram is a UML diagram with <Table> stereotypes. A GeoPackage abstract object model has been added as a potential future work item for a subsequent version of the GeoPackage specification. Subsequent non-SQL implementations thereof such as GeoJSON will depend on such an abstract object model.

### 5.7 Comment: GeoJSON example

1. Requirement: General
2. Implementation Specification Section number: 13.3 Manifest XML Schema
3. Implementation Criticality: Minor
4. Comments/justifications for changes

Recommend that a GeoJSON example be defined in addition to XML to support common web/mobile use cases. These may require a JSON profile of the OWS Context document.

#### 5.7.1 SWG Response

Thank you for your comment. The manifest table has been removed from the GeoPackage specification. Metadata such as OGC Context documents stored in the metadata table may be used to provide links to online sources of GeoPackage data. The metadata table metadata column is specified as not null, with mime types other than XML are now supported by a metadata table mime type column.

## 6 Comments: Edzer Pebesma 2013-02-06

**PART A**

**1. Evaluator:**

Institute for Geoinformatics, Univ. Muenster, 52°North
Edzer Pebesma, Christoph Stasch, Christian Autermann, Daniel Nüst
email: edzer.pebesma@uni-muenster.de

**2. Submission:**

OGC 12-128r1 Version: 0.2.0 - OpenGIS® GeoPackage Implementation Specification

**PART** B

### 6.1 Comment: Contents table max_y column type

1. Requirement: Figure 2
2. Implementation Specification Section number: 7
3. Implementation Criticality: Minor
4. Comments/justifications for changes

geopackage_contents: max_y should be REAL, not BLOB in Figure 2

#### 6.1.1 SWG Response

Thank you for your comment. Agreed. The table diagram has been revised. The table is now named gpkg_contents.

### 6.2 Comment: Non-XML Encodings

1. Requirement: General
2. Implementation Specification Section number: General
3. Criticality: Major
4. Comments/justifications for changes:

The document has a general mix of model and encodings. For example, manifest and metadata are bound to specific encodings (i.e. XML).

Instead, different encodings of a model such as JSON or plain text should be allowed and further defined in application profiles. Especially mobile systems, which are referenced in the motivation, are often not based on XML for good reasons.

#### 6.2.1 SWG Response

Thank you for your comment. A mime_type column to support formats other than XML has been added to the metadata table. A GeoPackage abstract object model has been added as a potential future work item for a subsequent version of the GeoPackage specification. Subsequent non-SQL implementations thereof such as GeoJSON will depend on such an abstract object model.

### 6.3 Comment: srid column values consistency

1. Requirement: General
2. Implementation Specification Section number: 7
3. Implementation Criticality: Major
4. Comments/justifications for changes

There are several srid entries in different tables of the model, e.g. geopackage_contents, geometry_columns, raster_columns, etc. It remains unclear, how these different srid entries are related to each other. For example, does an srid entry in the geometry_columns table overwrite the srid of the geopackage_contents entry? Does it need to be consistent? This needs to be clarified in the document.

#### 6.3.1 SWG Response

Thank you for your comment. The raster_columns table has been removed from the GeoPackage specification as part of the resolution of the DataColumnsTable discussion item, and the tables and column have been renamed, so there are now srs_id columns on only three tables.

The relationships among those three values are now specified by the following foreign key relationships:

- gpkg_contents.srs_id SHALL be the value of srs_id for a row in gpkg_spatial_ref_sys

- gpkg_geometry_columns.srs_id SHALL be the value of srs_id for a row in gpkg_spatial_ref_sys

### 6.4 Comment: Temporal properties

1. Requirement: General
2. Implementation Specification Section number: General
3. Implementation Criticality: Major
4. Comments/justifications for changes

Introduce facility for specific modelling of time.

Geopackage should accomodate infrastructure in the metadata tables that refers to time properties of records. For instance, just like the geometry_columns points out where geometries are found, a temporal_columns meta-data table should, if present, point out where the time information is to be found.

Can repeated measurements and time series be stored in geopackage?

#### 6.4.1 SWG Response

Thank you for your comment. The SWG agrees that temporal capabilities are important but thinks they should be deferred. The following potential future work item was added: "Future versions of this specification may add infrastructure to the metadata tables such as a temporal_columns table that refers to the time properties of data records.

### 6.5 Comment: SpatiaLite Reference Implementation

1. Requirement: General
2. Implementation Specification Section number: General
3. Implementation Criticality: Major
4. Comments/justifications for changes

Explain the role of the reference implementation. Why is that part of the specification? Some parts of the reference implementation seem to be normative for the specification.

What is the provenance for SpatiaLite? How large is the developer base currently? How does the project currently handle extensions offered by third parties?

#### 6.5.1 SWG Response

Thank you for your comment. The GeometryEncodingFormatSelection discussion item was resolved by selection of a SQL BLOB binary geometry format other than that used by SpatiaLite, so the GeoPackage spec is now based solely on SQLite and no longer depends upon or uses SpatiaLite as a reference implementation.

### 6.6 Comment: Relationship to other OGC standards

1. Requirement: General
2. Implementation Specification Section number: General
3. Implementation Criticality: Major

4. Comments/justifications for changes

The specification misses the relations to other OGC standards (e.g. O&M, SOS, WFS, WCS,WMS,WMTS, FilterEncoding). Some questions that come into mind:

- Should/can data modelled in O&M be stored in a Geopackage?

- Tiling: What is the relation to WMTS?

- Should services like a WFS also be able to return GeoPackages?

#### 6.6.1 SWG Response

Thank you for your comment.

1. Yes, with one qualification. Observations are GML Features conforming to the ISO General Feature Model, so they can be stored as vector features in a GeoPackage. But by default, GeoPackage only support geometries with linear interpolations between points; geometries with non-linear "curved" (bezier, bspline ...) segments require implementation of a registered extension.

2. The GeoPackage requirements for storing tiles from tile matrix sets in tiles tables were written to allow a GeoPackage to contain any set of tiles served by a WMTS. So a GeoPackage could serve as the back-end data store for a WMTS service.

3. The GeoPackage draft specification explicitly lists as potential future work:

- Future enhancements to this specification, a future GeoPackage Web Service specification and modifications to existing OGC Web Service (OWS) specifications to use GeoPackages as exchange formats MAY allow OWS to support provisioning of GeoPackages throughout an enterprise.

## 7 Comment: Doug Newcomb 2013-02-06

**Part A**

1. Evaluator:
   Doug Newcomb, USFWS, 919-856-4520 x14 doug_newcomb@fws.gov

2. Submission:

OGC 12-128r1 Version: 0.2.0 - OpenGIS® GeoPackage Implementation Specification

**Part B**

### 7.1 Comment: File System Capabilities

1. Requirement: 12
2. Implementation Specification Section number: Note
3. Implementation Criticality: Minor
4. Comments/justifications for changes

I would change "Some evolution in Mobile / Handheld Computing Environment file system capabilities will be necessary to allow such large files, e.g. in EXT file systems instead of FAT32 ones that limit file sizes to 4 GB." to " Some evolution in Mobile / Handheld Computing Environment file system capabilities will be necessary to allow larger file sizes than 4GB, the current limitation of FAT32." There are many filesystems,

such as NTFS, UFS, and EXT4 , as well as XT that allow for larger file sizes than 4 GB. FAT32 should only be named because of it's known limitation in this regard to avoid the appearance recommending a specific filesystem.

#### 7.1.1 SWG Response

Thank you for your comment. The SWG replaced the "Some evolution ..." sentence with "The maximum size of a GeoPackage file is about 140TB. In practice a lower size limit may be imposed by the filesystem to which the file is written. Many mobile devices require external memory cards to be formatted using the FAT32 file system which imposes a maximum size limit of 4GB."

### 7.2 Comment: Serverless database implementation

1. Requirement: 44
2. Implementation Specification Section number: General
3. Criticality: Major
4. Comments/justifications for changes:

It is vitally important that this standard be based on a robust, fully documented serverless database implimentation. The choice of Sqlite with the libspatialite extension as the reference implementation is an excellent choice. This option combines the widely used , well tested, and well documented public domain sqlite database with the proven open source libspatialite spatial database extension. A standard is not open without an open source reference implementation

#### 7.2.1 SWG Response

Thank you for your comment. The GeometryEncodingFormatSelection discussion item was resolved by selection of a SQL BLOB binary geometry format other than that used by SpatiaLite, so the GeoPackage spec is now based solely on SQLite and no longer depends upon or uses SpatiaLite as a reference implementation. After proof of concept via a "mini-plugfest" between three GeoPackage SWG members implemented the selected SQL BLOB binary geometry format. OSGeo and Luciad have said they will provide open source implementations.

### 7.3 Comment: libgeos

1. Requirement: 45
2. Implementation Specification Section number: General
3. Implementation Criticality: Major
4. Comments/justifications for changes

libgeos is a compact and robust vector library and is an excellent choice for maximum flexibility

#### 7.3.1 SWG Response

Thank you for your comment. The GeometryEncodingFormatSelection discussion item was resolved by selection of a SQL BLOB binary geometry format other than that used

by SpatiaLite, so the GeoPackage spec is now based solely on SQLite and no longer depends upon or uses SpatiaLite as a reference implementation. So the GEOS lib may be used by some implementation of a GeoPackage, but is completely optional.

### 7.4 Comment: Proj4

1. Requirement: 46
2. Implementation Specification Section number: General
3. Implementation Criticality: Major
4. Comments/justifications for changes

Proj4 has a long development history as a lightweight, robust , and versatile projection library and is well suited for this task.

#### 7.4.1 SWG Response

Thank you for your comment. The GeometryEncodingFormatSelection discussion item was resolved by selection of a SQL BLOB binary geometry format other than that used by SpatiaLite, so the GeoPackage spec is now based solely on SQLite and no longer depends upon or uses SpatiaLite as a reference implementation. So the Proj4 lib may be used by some implementation of a GeoPackage, but is completely optional.

## 8 Comment: Chris Holmes 2013-02-07

**Part A**

1. Evaluator:

2. Submission:

OGC 12-128r1 Version: 0.2.0 - OpenGIS® GeoPackage Implementation Specification

**Part B**

### 8.1 Comment: Don't like the way GeoPackage spec is written

1. Requirement: General
2. Implementation Specification Section number: General
3. Implementation Criticality: Major
4. Comments/justifications for changes

First I must apologize for the huge amount of text I am going to shove in to this first comment. I wrote up most all my geopackage responses before looking at this response format, and I'm not sure how it fits in. But I will just leave the bulk of the general comments here, and then repeat section 'B' for the more focused comments.

So why all these comments? The new GeoPackage specification contains some ideas with the potential to be one of the best things to come out of the OGC in many years. And I'd

really like to see it become a successful specification, reversing the trend of less and less clear, relevant, and implementable standards.

A bit of background - I've been peripherally involved in the OGC since 2002, primarily as an implementor of specifications. I did the first open source implementation of the WFS specification, and have guided OpenGeo through a number of OWS initiatives, including making GeoServer the reference implementation for WFS 1.0, 1.1 and WCS 1.1. My organization has never been able to afford to attend in person OGC meetings and have a big hand in influencing the specifications. But we have faithfully implemented the specifications as much as possible, and tried to give feedback to those we are connected with. But it seems challenging to have a voice without a major investment in time and money to make all telecons and in person meetings.

In the last five years our developers have become increasingly frustrated with OGC standards, as they have become harder and harder to implement, with less relevance to our users and customers. We now only implement the new OGC standards when a customer pays us to, as our investment is better spent on other improvements. We've grown increasingly frustrated with the results of the OGC process, and when we've tried to help we've found it costly and time consuming. The quality of OGC specifications has gone down in recent years, while our needs have been much better met by 'open source' style specifications like GeoJSON, TMS, MBTiles and more. We also have a far easier time contributing to those standards.

I've engaged OGC staff members on this, and one consistent message is that we need to get more involved, to push harder, to explain our viewpoint. This set of comments is my attempt at doing that. Though much of it may come across as negative keep in mind that it's an attempt to help, as I fully believe in open standards, and I think the OGC has done an amazing job of getting organizations around the world to speak standards and require them in their solutions. But there is a real threat to the OGC, which is that if the quality, approachability and implementability of specifications goes down too far then people will start finding alternatives to open standards. This has already been happening, as I constantly have to defend our continued implementation of OGC standards to a number of allies.

Much of what the OGC has done in recent years is to move to the edges, helping information communities standardize what they need to. Which is great, and makes sense. But there needs to be a strength in the geospatial core that all will use. The vector portion of the GeoPackage specification has amazing potential, to fill a major general hole in the geospatial world. A great, web-friendly widely accessible transfer format could fill a number of needs, and get quick adoption. It is one of the biggest opportunities for the OGC, which is why I am sounding in now, as I want to be sure it fulfills its potential with maximum adoption. The tile portion of the specification could also do quite well, meeting a real need. The third area with major potential for standardization is a CSS type successor to SLD, as there are a number of implementations that could do well to be standardized. But for the most part these opportunities are rare, when many people are searching for a standard solution.

* General comments of OGC specifications

Before I start in on specifics of comments and general strategies for GeoPackages I want to write a bit more broadly about where I'm coming from. In general the circles I spend most of my time in are very negative on OGC specifications. At first they were viewed as a necessary evil, but worth doing. In the last 2-3 years most people are questioning if they are even necessary, and many have been concluding that they are not, moving ahead with open source code complemented by lightweight specifications. This audience is more of 'the cool kids', the open source crowd, the younger developers. But it should be remembered that they will be the ones coding the future, and if they don't see the value in OGC standards then they just won't implement them.

Note that this doesn't mean they don't see the value in openness, they do, and indeed most of them insist in working in a totally open source way - code and standards. And they will create standards when they are needed. My ideal would be to combine their innovation with the amazing work the OGC has done in getting acceptance of standards at the highest levels of government and industry. There are two main things to change. The first is easier, which is just making the specifications more accessible and relevant, easier to implement a valuable core. The second will be more challenging, and will likely require some deep shifts in how the OGC approaches creation of standards. I will focus primary on the first concern here, though parts of the second, the current culture around creation of specifications, will likely creep in.

The first concern is well articulated by Justin Deoliveira. He's probably spent more time implementing OGC software than all but a handful of developers in the world. He did most all of WFS 1.1 and 2.0 in GeoServer, built the only truly general purpose GML reader that I know of (able to parse the schemas on the fly and turn them in to a real feature model), worked extensively with WMS, WCS and WPS, and helped implement the core complex feature model. I asked him about the geopacakge spec, and he said:

'Yup. Reading this makes me feel warm and fuzzy inside: http://mapbox.com/developers/mbtiles/

While reading the geopackage spec makes me want to run for the hills.'

I believe it's worth explaining why I believe he says this, as it's relatively easy to dismiss it for the wrong reasons. The same comments could just as easily be said about GeoJSON or TMS.

First, it's a web address that can be read right away. It doesn't require downloading a pdf or word document, or accepting some click through agreement. Web native developers prefer to read things directly on the web. It's a subtle thing, but even a pdf doesn't 'feel' write, it speaks to control, not openness. And firing up Word is annoying. It puts a higher

barrier to entry, and will give pause to a developer deciding whether a spec is worth implementing and then looking to suck down information after deciding it is.

Second, there's no boilerplate. I don't have to scroll through 4 pages of title and table of contents to learn what it's about. And after those 4 pages there's more cruft with a preface, a forward, and an introduction, lists of contributors, past versions.

Third, it doesn't open like a marketing document telling me all the problems it will solve - that it will enlarge the market and become the standard. MBTiles opens with exactly what it does: 'MBTiles is a specification for storing tiled map data in SQLite databases for immediate usage and for transfer.' It doesn't try to convince me it will become _the_ standard, it just tells me what it does so I can decide for myself if it's useful.

Fourth, it tells me the core information and no more. It's 3 pages, communicating to me that it can't be very hard to implement. GeoPackage actually should not be hard to implement if I've got Spatialite, but it took me half an hour of reading to realize that. A key thing here is that MBTiles provides links to what it builds upon. It doesn't replicate the TMS specification, it just links to it for people to read (this is where being on the web shines, as it doesn't require leaving word or acrobat to get more information). After the link it tells you how it's different, that it's constrained to web mercator. By contrast the geopackage vector section lists every single sql geometry operation. And then has a matrix telling me which of 4 other specifications implements it. One could fairly easily just say that it's based on 13249-3 and adds Add/Drop GeometryColumn operations (and perhaps takes some away?) The general point is that the specification should present the user with exactly the information they need to implement it, no more and no less.

Fifth, it's modular. It doesn't require UTFGrids. And it doesn't include them in the main document in some conformance class that may be optional. It is an add on that also stands alone, so it can live and die on its own merits. If it was in the main document then I'd be confused as to if I _had_ to implement it - it being an add on communicates to me that it's not part of the core. The GeoPackage document will dump large amounts of information (like trigger statements in sql), and then some sub-clause will let me know that it's optional. It doesn't respect the reader, it makes the reader pay super close attention to every single detail.

Another big issue is the perception of how easy it is to contribute. Even to comment during the RFC period takes filling out some precise template. Not that it's a big burden. But any modern software developer likely knows how to comment or contribute on github like the back of their hand. Just by having the spec on github people know that it's meant to be forked and adapted, innovated with. The infrastructure communicates that to developers, while the OGC infrastructure communicates that they have to go through some huge process to give their feedback - that it's an 'experts only' place, a hermetic club that can spend their time writing specifications.

Personally I've also always had no idea if my feedback has any effect in the OGC. There's no feedback looping letting me know that what I contributed was valuable, because it rarely emerges, and even more rarely in an identifiable way. I found out much later that some of what I've said in the past has been talked about a lot. But since I have no feedback loop there's little to continue to draw me. And joining the list to discuss it requires being a member, and most critical decisions get made on the phone, not the list. I can't easily see the discussion that lead to the spec. (Admittedly mbtiles isn't great in this regard, but all the historical changes are easily followed line by line. A much better example of open discussion is geojson - I can see all the decisions that lead to the implementation). I never feel like I could spend a bit of time and potentially contribute something valuable, I have to be all in on the specification process. With open source we have many developers who clearly have other concerns, but they're able to make valuable contributions without spending much time.

The one valid criticism that can be made on the mbtiles spec is that it doesn't cover all needs. It's too simple. I agree there is a good bit of truth to that, but I think it's a criticism that is used to overrule all other considerations. There's an attitude that if a spec doesn't cover every imaginable need than it's not worth it. Though this starts to get to the deeper cultural shifts needed - I'm trying here to limit discussion to the general accessibility of specs.

So what are the general lessons to be drawn from this comparison? The more superficial ones are to just publish specs that feel like web natives, that can be read online, that don't have lots of boiler plate, that get to the point. That have easy infrastructure for people to comment in the way want, to have conversations on blogs in iterative feedback loops. This is probably the easier part.

The deeper thing for me is the attention needed to understand the specifications. The GeoPackage spec counts an audience that is absolutely committed to implementing the standard. Who likely got assigned as part of their job to do it. Indeed the audience seems to be other specification writers, not developers who want to implement readers or writers for the standard. The mbtiles spec communicates some concepts to help people solve their problems. A big part of that is that MBTiles solves exactly one problem. GeoPackage attempts to solve all problems. And it's more concerned with its conformance rules than it is with communicating what an implementor needs to know.

I hope the above comments are helpful for some general ideas on how to make the specification more 'implementable'. To increase the audience who may be interested in helping interoperability. Some will probably be hard for the OGC to implement. But the thing I believe will help the most is to involve actual implementors of production ready software before a specification goes to 1.0. At OpenGeo we've implemented a lot of specifications, likely as many as all but a couple organizations in the world. And we've found that in recent years there are even more 'ideas' added to the specifications that have no true production ready working code against them. Past the surface accessibility this is the thing that has become clear to us as implementors - there is less quality control at the core of the standards process. There are things that are not well thought out, that don't

come from a variety of perspectives. And this is incredibly important for the core geospatial specifications. GeoPackage will be a much stronger specification if time is given for GDAL/OGR and GeoTools implementations, where the developers can feed back what to improve to make it easier to implement. And the side effect would be that 80% of all production deployed geospatial software would understand GeoPackages, as those two libraries are easily included in that number of systems.

#### 8.1.1 SWG Response

Thank you for your comment. With Respect To (WRT): a web address that can be read right away

CMT: A web page is convenient for developers, but of questionable immutability for a standard unless it is an informative replication of an immutable document. I hate citing web pages as normative references because there is absolutely no assurance the page won't disappear or the contents won't change.

WRT: boilerplate ... 4 pages of title and table of contents .. a preface, a forward, and an introduction, lists of contributors, past versions

CMT: These are required by OGC standards document templates; changing them is not within scope for a SWG. ISO and most national standards bodies use similar formats.

CMT: Perhaps OGC needs to develop a template for a "standards marketing web page" that is required to be created to promote each proposed standard.

WRT: opens with exactly what it does

CMT: The GeoPackage? Spec is constrained by OGC document templage guidance to "open" in the scope clause. My first run-on sentence in the scope clause clearly needs to be rewritten. See RewriteScopeClause.

WRT: geopackage vector section lists every single sql geometry operation

CMT: The SWG has discussed removing these lists or moving them to an appendix. Removing them makes the document much shorter, but has the disadvantage of requiring the reader to access other standards documents to obtain the definitions. Since 3 of these standards documents are published by ISO, that means the reader has to purchase them. Moving them to an appendix does not make the document any shorter, but is more reader-friendly. See GeometryFunctionsAnnex.

WRT: no feedback looping

CMT: The SWG will provide response to all comments on the GeoPackage? spec. The SWG cannot speak for the TC or PC.

WRT: can't easily see the discussion that lead to the spec

CMT: I agree that this has been a deficiency of most past spec development. The GeoPackage? SWG is trying to capture all discussion that leads to spec changes in wiki discussion pages such as this one under Discussion Items.

WRT: dump large amounts of information (like trigger statements in sql), and then some sub-clause will let me know that it's optional

CMT: Various schemes to reorganize the spec document were discussed during OWS-9, including moving the trigger definitions to an appendix. Doing so would shorten the main body of the spec, but would require readers to flip back and forth between the table definitions and trigger definitions to understand all the data content constraints involved.

The OWS-9 cosensus was that doing this was not of much benefit. The SWG will revisit the issue. See SQLTriggersAnnex

WRT: it makes the reader pay super close attention to every single detail

CMT: Is there any other way to build an implementation that conforms to a specification?

WRT: the perception of how easy it is to contribute

CMT: Changing OGC policies and procedures is out of scope for a SWG. OGC Staff has a copy of these comments. I will email them wrt non-SWG issues.

WRT: GeoPackage? attempts to solve all problems

CMT: GeoPackage? scope was initially all geospatial data types. Routes and coverages were soon deferred to future versions of the spec.

WRT: more concerned with its conformance rules than it is with communicating what an implementor needs to know

The OAB and OGC Staff have been making conformance to the ModSpec a key issue for all SWGS. The conformance rules form the basis for abstract and executable test suites. So they in fact communicate to an implementer what will be tested, the first step in test driven development.

WRT: GeoPackage? will be a much stronger specification if time is given for GDAL/OGR and GeoTools? implementations, where the developers can feed back what to improve to make it easier to implement.

CMT: Agreed. Frank Warmerdam has said he's looking into GDAL/OGR implementation.

CMT: But if you don't have time to attend OGC telecons, and you now only implement specifications that your clients pay for, when would these implementations ever get done?

**Proposed Resolution**

1. see RewriteScopeClause
2. see GeometryFunctionsAnnex
3. see SQLTriggersAnnex
4. IP/ TC / PC / BOD to amend OGC policies and procedures

Thank you for joining the GeoPackage SWG and becoming an active voting member to more directly advocate for your point of view on appropriate contents for the specification.

### 8.2 Comment: GeoPackage specification granularity

1. Requirement: General
2. Implementation Specification Section number: General
3. Implementation Criticality: Major
4. Comments/justifications for changes

So the major thing I'd like to address in this comment is the granularity of the GeoPackage specification. I believe it should be split up in to 5 succinct specifications,

that only contain the core conformance classes. Additional extensions / non-core conformance ideas should be in extended specifications.

There was a line in the introduction (which comes after the forward and the preface) that hits on what is to me a big problem: 'An application that accesses a GPKG will make use of the GPKG capabilities it requires; few if any such applications will make use of all GPKG capabilities.'

I think this is actually just rude to any implementor. Even though an implementor, by the spec's own admission, is probably going only going to use a part of the spec, they are more or less forced to read and understand the whole spec. Because there might be something relevant to them. The specifications authors should work to communicate to implementors exactly what they need to know. I think there's improvements towards that which can be made in each of the major sections, but I'll go in to those individually.

I believe there are actually five potentially great specifications in there. I also believe they are of varying quality and real world relevance. By putting them in one single specification it brings down the overall quality, as some things will be more thought out and tested and somethings will be less so. If they are in separate specifications than each can evolve on its own. And each can also live and die in the market on its own. Maybe that way of doing metadata just doesn't work well. Then the metadata geopackage spec can be upgraded, without having to touch the others. Maybe people just don't end up storing metadata. Maybe they don't use the manifest. Keeping things independent lets things evolve more easily.

It also just makes for good architectural design practices, keeping things modular. The OGC has made a lot of effort to make specs modular. But in my opinion there is a large difference between having a core spec that one knows must be implemented and some optional specs that add to it versus a big long spec with some conformance table that I have to read to see what's optional or not. With a set of complementary specifications I can pick and choose the one most important to me, and later pick up the others when they become relevant.

By splitting up in to 5 and reducing some of the extraneous information the geopackage specification could get to 5 really nice mbtiles-like specifications. Implementors would be able to grok the main concepts in minutes and get started on implementing, likely taking little time to get an interoperable standards compliant implementation in their software.

The five specifications would be vectors, tiles, rasters, metadata and manifest. I do like the idea of a manifest, but I don't feel it should be mandatory, at least not the full xml document. One should be able to implement one of the component pieces and just a snippet of manifest to help identify it properly. And each specification should contain _only_ the core conformance classes. The additional conformance classes should be in fully separate specifications, to fully and quickly communicate to implementors what they _must_ do and what is some useful ideas if they want to do more.

Though I am a bit biased, as I have a bigger vector background, I believe a concise gpkg-vector spec with some improvements could be wildly successful. The market has been wanting this. I have some thoughts on how I believe it could be specified better, but I'll articulate those in its own section. But it should be able to stand alone, with implementors just reading that one small spec and starting to create and consume vector geodata.

And each of the other pieces could be quite valuable on their own, solving smaller use cases. Together they should describe a whole package that can be shipped in one sqlite database file. But an implementor shouldn't need to understand all the parts to just do what they want. Unless they are making one of the few applications that requires all of it. But they can build up to there with each specification.

#### 8.2.1 SWG Response

Thank you for your comment. According to the ModSpec, a complete specification "Base" consists of a "Core" and "Extensions". "Profile"s are subsets of the "Base". Splitting up the GeoPackage spec as suggested would need "Core" and "Base" specification documents in addition to the five suggested "Profile" documents. This approach was taken for the WCS v2.0 spec. I find that I need to have two or three documents open at once to understand one protocol binding for WCS v2, whereas one document was sufficient to understand one protocol binding of WCS v1. I don't think creating seven separate specifications would make GeoPackage any easier to understand or implement. I'd suggest that we discuss reorganizing the specification document instead. The SWG voted to retain a single specification document and continue on the path we have taken since SpecificationDocumentOrganization was decided upon; separate clauses for core and extension requirements with as much detail as possible moved from the main document body to annexes.

### 8.3 Comment: SpatiaLite dependency

1. Requirement: General
2. Implementation Specification Section number: 9
3. Implementation Criticality: Major
4. Comments/justifications for changes

While I believe the vector portion of the geopackage specification has the most potential there are a number of core issues that need to be addressed. I believe the biggest issue is the specification depends directly on Spatialite. The beauty of using Sqlite is that it ships with practically everything, so an implementor can easily read it or even write to it without having to install any additional libraries. It ships on android phones, it ships with browsers, etc. Spatialite lib does not ship with much at all. So it's an additional dependency that I believe will greatly hinder adoption, since it won't be readable except by people who have taken the time to include geospatial libraries.

Now what I believe could work well is specifying a blob geometry. This could just use spatialite's nicely defined binary spec, or it could be something else. Sqlite is great in that if an extension is not installed then it just returns the data as blobs. So an implementor who does not have access to spatialite on their sqlite could still read a geopackage vector

file and know how to unpack the blob and use that geospatial coordinate. It doesn't matter that sqlite has no geometry blob already defined, it's a system that doesn't depend on needing that in the main specification. If it was just a geometry blob from sqlite then implementors wouldn't have a spatial index or all the spatial operations in simple features for sql. But many cases won't need that. A mobile device might just have 50 points, where a spatial index is useless. They may just want to read and write points. Additionally other libraries may offer some spatial capabilities on their own, and could do more but not have to figure out spatialite lib dependency management. Using spatialite's binary definition as the blob could work quite well, since that system and all the code already written to leverage it could use and take advantage of any geopackage vector geometry blob.

#### 8.3.1 SWG Response

Thank you for your comment. The ContainerOnlyOrTieredArchitecture discussion item was resolved by selection of a two-tiered architecture. A "level 1" geopackage implementation can read an existing GeoPackage or write a new GeoPackage without using any libraries if it does not intend to use SQL, and may read or write a GeoPackage via SQL using only the SQLite lilbrary.

The GeometryEncodingFormatSelection discussion item was resolved by selection of a SQL BLOB binary geometry format other than that used by SpatiaLite, so the GeoPackage spec is now based solely on SQLite and no longer depends upon or uses SpatiaLite as a reference implementation, so there is no longer any lib dependency there.

A "level 2" geopackage implementation that can update an existing GeoPackage will depend on a minimal library implementation.

### 8.4 Comment: Binary format vs API

1. Requirement: General
2. Implementation Specification Section number: 9
3. Criticality: Major
4. Comments/justifications for changes:

I've been informed that there is some active discussion of this on the GeoPackage SWG list, but I wanted to sound in with my opinion on standardizing an API versus standardizing the geometry format. I strongly feel the only part that should be standardized is the geometry format. It should be a single binary format. If desired the API could be standardized in an optional specification, that lays out all the table names and triggers. The core standard should just be the geometry blob, the underlying sqlite file format. That's what will get passed around. That's what needs to be read. I believe it's ok to specify it as the binary that results from sqlite with geometries stored in a certain way. An alternative would be to fully specify the binary structure of sqlite. But just specifying the table structure could lead to unanticipated results. One of our developers suggested doing an java implementation of the specification. As specified this would mean we'd match the table structures and api's, like with jts plus an embedded database like h2. Like https://github.com/jdeolive/geodb this could match the specification, except for the parts that say it requires libspatialite. But it'd result in a totally different binary format. I've heard there is discussion of multiple binary geometry formats, and just

'focusing on the api'. I believe this is also the wrong approach. This will greatly decrease interoperability, as people will make readers and writers for the one that makes sense to them. And then getting a geopackage vector format may not actually be able to be read.

As a standards organization the OGC best serves its users and members by choosing exactly one format. This has been an increasing problem in recent specifications. The original get vs post bindings made some sense, one offered an easy way to do things through the browser, the other often had more options and possibilities. But specifications like GeoSynchronization service are frankly ruined by 4 different bindings. As an implementor you have no idea which to do. They don't even recommend one as the best way. So to implement the spec at all you have to do all of them, or risk that the other end won't be able to read it. At OpenGeo we experienced this directly with the WMTS specification. We just had funding for one of the bindings, and our client funded a front end for it that read the other binding. So even though they had funded implementation of the specification since they didn't fund it all or coordinate on which to have both do they got nothing. Specifications should be one standard, not 4, or even 2. So please just pick one geometry format, and standardize on that. The API to access it and maintain consistency should be a separate specification that builds on it (and it should not be included in the main specification as optional conformance classes)

#### 8.4.1 SWG Response

Thank you for your comment. The ContainerOnlyOrTieredArchitecture discussion item was resolved by selection of a two-tiered architecture. A "level 1" geopackage implementation can read an existing GeoPackage or write a new GeoPackage without using any libraries if it does not intend to use SQL, and may read or write a GeoPackage via SQL using only the SQLite lilbrary.

The GeometryEncodingFormatSelection discussion item was resolved by selection of a SQL BLOB binary geometry format other than that used by SpatiaLite, so the GeoPackage spec is now based solely on SQLite and no longer depends upon or uses SpatiaLite as a reference implementation, so there is no longer any lib dependency there.

A "level 2" geopackage implementation that can update an existing GeoPackage will depend on a minimal library implementation.

### 8.5 Comment: Optional Conformance Clauses

1. Requirement: General
2. Implementation Specification Section number: 9
3. Implementation Criticality: Major
4. Comments/justifications for changes

All optional conformance classes should be moved to a separate specification or appendix. I realize this is not the current OGC way of doing things, but I believe it greatly improves readability. The specification should also not spend so much time listing every single geometry operation. Especially since it's an exact repeat of the capabilities of spatialite, without saying that. It should be up front that spatialite is required and the only working implementation (though I don't think it should be required, see other comments). It honestly took me 20 minutes to figure out that this was just a standardization of

spatialite. Though I think standardization of sqlite using spatialite geometry format is far more powerful I do think standardizing on spatialite could do a lot. But in the spirit of having the specification communicate just the right information it should say that up front.

The specification is also made way more confusing and unreadable with the trigger statements scattered through out. These should move to a 'geopackage vector consistency api' standard, not cluttering up the main core standard. Most people will likely just use spatialite, so won't need to bother with reading the whole thing, they can just trust that they're using the standard reference implementation and it's taking care of all the details. So all the constraint checking can be in a side best practices paper on using spatialite to implement the core. Or it could be a standard on the 'api' of all the sql statements and table structures that can make it so.

#### 8.5.1 SWG Response

Thank you for your comment. The draft GeoPackage specification has been reorganized to move all optional conformance classes to a separate clause and SQL Triggers to an annex as resolution of the SpecificationDocumentOrganization and SQLTriggersAnnex discussion items.

### 8.6 Comment: gB and gS storage architectures

1. Requirement: 18,19
2. Implementation Specification Section number: 9.4
3. Implementation Criticality: Minor / Editorial
4. Comments/justifications for changes

The "“gB” and “gS” architectures' are super confusing and don't seem to add any value. I think I understand this stuff pretty well and am still not sure of the differences and when I might want to use one or the other, or if I even have the possibility in spatialite. I'd say just pick gB for people, standardize on it for now. If core tech improvements make the other possible and more attractive then put it in a future version of the spec. Reading that paragraph feels like an engineering report on the possibilities, not a spec providing guidance to people.

#### 8.6.1 SWG Response

Thank you for your comment. The draft GeoPackage spec has been revised to remove references to the SF/SQL gB and gS storage architectures.

### 8.7 Comment: Spatial Index Requirement

1. Requirement: 25
2. Implementation Specification Section number: 9
3. Implementation Criticality: Minor
4. Comments/justifications for changes

I don't think that a spatial index should be a requirement. This goes a bit back up to the core points above about just sqlite. But indexing should be an optional piece, not in the core spec, at least not initially. Shapefile is actually a good example, where a standard index file evolved. But we should let implementors experiment with indexing schemes and then standardize what works best. Spatial indexing should be an extension to the core vector geopackage, not a required part of it. I realize this is a different philosophy than most OGC specs, but I think it's something that can make the specs much better. Keep them focused on meeting the most needs, and put the more obscure use cases in extensions to the core. I admit that many will want a spatial index. But as far as I know the driver for geopackage is mobile devices, and many of those will just be a small number of point features that a user is editing, where a spatial index doesn't add much at all.

#### 8.7.1 SWG Response

Thank you for your comment. The ContainerOnlyOrTieredArchitecture discussion item was resolved by selection of a two-tiered architecture. A "level 1" geopackage implementation is a required core piece. A "level 2" geopackage implementation including a minimal library that includes functions needed to support implementation of spatial indexes is an optional extension piece.

### 8.8 Comment: Don’t discuss parts of SF/SWL specs not used in GeoPackage

1. Requirement: General
2. Implementation Specification Section number: 9.3
3. Implementation Criticality: Minor / Editorial
4. Comments/justifications for changes

On page 20 the specification says: 'SQLite does not have separate catalogs or schemas, so the f_table_catalog and f_table_schema columns are meaningless in an SQLite GPKG container....' That whole page again reads more like an engineering report. It's an evaluation of the simple features for sql specification against the capabilities of sqlite. The spec doesn't need a list of the non-relevant ones and the reasons they aren't relevant. It just needs the list of ones to implement. Again, this comes down to the fact that we actually _want_ sqlite (with or without spatialite lib, geos lib and proj lib) to be the only implementation of it, because we want that file format it emits to be standard and shared across systems. So we don't need to talk philosophically about how it ideally would implement simple features for sql or sqlmm. Once it can then we just update the spec for a new version that understands that.

#### 8.8.1 SWG Response

Thank you for your comment. The draft GeoPackage spec has been revised to remove references to the f_table_catalog and f_table_schema columns. architectures.

### 8.9 Comment: Redundant discussion of SpatiaLite

1. Requirement: General

2. Implementation Specification Section number: 9.6
3. Implementation Criticality: Editorial
4. Comments/justifications for changes

On page 37 'Note3: SpatiaLite is currently the only known vector feature store based on SQLite that meets the specifications documented above for vector features and SQL geometry routines.' This line and the following paragraph seem entirely unnecessary, since REQ 44 on page 36 says that it shall be a 'SQLite database with libspatialite loaded to provide SpatiaLite extensions of version 3.0.1'. The spec doesn't just specify the routines, it requires the actual library. Which again, I don't think is a bad thing, it should just be clear up front and shouldn't waste a whole bunch of people's time repeating all the capabilities that spatialite offers and how to set up its triggers. That should be a complementary best practices or engineering paper.

#### 8.9.1 SWG Response

Thank you for your comment. The GeometryEncodingFormatSelection discussion item was resolved by selection of a SQL BLOB binary geometry format other than that used by SpatiaLite, so the GeoPackage spec is now based solely on SQLite and no longer depends upon or uses SpatiaLite as a reference implementation. All normative references to SpatiaLite have been removed from the draft spec.

### 8.10 Comment: Multiple Implementations Bad

1. Requirement: General
2. Implementation Specification Section number: 9.6
3. Implementation Criticality: Major
4. Comments/justifications for changes

On page 36, there's a microsoft word comment on spatialite 'The GeoPackage Software Working Group (SWG) should discuss further revisions to the specification to enable development of other reference implementations.' This seems to me to be a very, very bad thing since then we will end up with multiple binary files for databases that meet the requirements (though obviously those requirements would have to be relaxed from requiring lib spatialite). And users won't know which one they're supposed to use, the one produced by C or Java or some other tool. There _should_ be just one reference implementation, and indeed it should be explicitly specified. Or if that doesn't fly then just specify the binary file format. That could then be read by other implementations, and it wouldn't lead to many different formats that all have the table structures, geometry operations and constraints done right but store the data differently.

#### 8.10.1 SWG Response

Thank you for your comment. Someone in OWS-9 suggested we should consider alternative reference implementations, so I added this comment to the spec draft to assure the issue would be discussed. I agree with Chris that there should be only one reference implementation. The comment and all normative references to SpatiLite have been removed from the draft spec.

the comment and all normative references to SpatiLite

### 8.11 Comment: Tiles should not depend on SpatiaLite/RasterLite

1. Requirement: General
2. Implementation Specification Section number: 10
3. Implementation Criticality: Major
4. Comments/justifications for changes

I spent less total time on this section, so don't have as many specific comments. But my general sentiment from vectors is even more true here. The tile part of the specification should be done with pure sqlite, no dependency on spatialite/rasterlite. And it should just focus on tiles, not on a view structure of raster data. I think there's actually a great logical way to make this in to two complementary specifications.

The tile portion should be just some small extensions on MBTiles to handle multiple projections, which I understand is the big immediate need that MBTiles is not covering. But the specification should strive to cover exactly that need, not 5 more future needs. At some point those should be covered, when it's clear that the core is working quite well and people naturally extend it to try more things. Once a few people have tried new things then the core can be extended. Ideally the tile portion of the specification is super legible to anyone who has implemented MBTiles, they'd be able to grok in a couple minutes that they just have to tweak a few things to handle additional projections. It should be nice and simple, no more than 5 pages. Ideally it could eventually merge with MBTiles, in an MBTiles 2.0 once they see the need to handle more projections. Or even more ideally they just point at the geopackage tile spec as the new way of doing things. But that will _only_ happen if it's super simple, like MBTiles, and then extends for just projections.

And again, it is essential this core is just sqlite. It should be able to work on any existing sqlite system, not requiring spatial extensions, and not requiring lots of tables and triggers. Tiles are quite simple and MBTiles has proven this is possible. This is a great opportunity for the OGC specifications to simply extend what has already gotten great uptake. A counter example of where the OGC has done a poor job of this is WMTS. The TMS specification was nice and simple, and WMTS introduced way too much complexity. It took our GeoWebCache developers 10 times as long to implement, as it added lots of complexity to handle use cases most don't care about.

The functionality that includes views on raster data though seems like an ideal extension. It offers more capabilities on top of the core tiling, representing the tiles as SQL Views on top of the core raster types.

#### 8.11.1 SWG Response

Thank you for your comment. The GeoPackage spec now has no dependency on spatialite/rasterlite. Metadata tables for individual rasters are now separate from those for

tiles. There is now only one tile-specific metadeta table - tile_matrix_metadata. Normative text on that table and user data tiles tables is now only four pages long.

### 8.12 Comment: Don’t mention RasterLite

1. Requirement: General
2. Implementation Specification Section number: 9.6
3. Implementation Criticality: Editorial
4. Comments/justifications for changes

It would be good to remove all reference to rasterlite if it is not actually required. I walked away with the impression that it was, though a Geopackage SWG member pointed out to me it wasn't. It is so heavily featured though that one assumes it must be. I think that type of information belongs in an engineering report, not in the spec proper. With one exception, and that would be if it could save lots of space by just referencing spatialite's raster operations. Also all the optional conformance stuff, triggers and the like, should go in a separate specification, to keep the core clear. The specification should just describe the table structure, and then a best practices can include a list of triggers to help maintain the consistency of that table structure.

#### 8.12.1 SWG Response

Thank you for your comment. References to RasterLite have been removed from the draft GeoPackage specification document.

### 8.13 Comment: Metadata section

1. Requirement: General
2. Implementation Specification Section number: 11
3. Implementation Criticality: Major
4. Comments/justifications for changes

The metadata section of the specification is actually quite well done. It is not required, and nice and short. It depends on other specifications, and for the most part just references them. The bit of listing scope codes is actually quite useful, citing where they come from, and then listing exactly the ones that are relevant for this part of the specification. I think the metadata section could be a great stand alone specification, that complements any of the other geopackage specifications. Each can refer to the metadata spec, as the recommended way to add additional metadata.

One thing that could be nice is a recommended, though not required, metadata format. Like iso19139, something widely used. But let people know it can be others. But someone who doesn't know otherwise but wants to add some metadata can have a recommendation of which to use. Those who care about a particular metadata format can override that. Europe might additionally specify that geopackages from government sources should all include INSPIRE metadata documents.

Reiterating what I said in other sections I do think the listing of triggers should be removed from the specification proper. The specification should just list what a properly implemented table looks like. A best practices paper can list the sql triggers.

#### 8.13.1 SWG Response

Thank you for your comment. The GeoPackage SWG has voted that the version 1 of the GeoPackage specification will be contained in one document. A new footnote 2 has been added to clause 2.4.2.1.2 Metadata Table 11 recommending ISO 19139 for general-purpose description of geospatial data at the series and dataset metadata scopes.

### 8.14 Comment: Manifest in separate specification

1. Requirement: General
2. Implementation Specification Section number: 13
3. Implementation Criticality: Major
4. Comments/justifications for changes

The manifest document should also be its own specification. Most all other ones will refer to it. But I believe the others should be designed in such a way that they don't _require_ it to be there to do anything useful. It should just provide additional information about where a geopackage might fit in to a broader context.

I need to spend some more time thinking about it, but I think having it be an xml document is weird. Mostly in regards to the expectations of the rest of the specification. The rest is all about an API for developers to access, a nice sql api to do a lot. But the xml document is more of a raw thing that requires libraries to read it. We should be providing developers a consistent interface. One way to do that is to ship a geopackage api that includes good xml parsing stuff. And then documents the calls to get at the information in the manifest. Alternatively the manifest information could be represented as tables, so that it is a consistent API. But a big xml document smashed in to the spec makes for an inconsistent api, and greatly raises the barriers to implementation. It requires developers who are fluent in both xml access and sql. It can be argued that both are fairly standard skills for developers. But most are likely stronger in one or the other, so you end up being as hard as their biggest weakness. I've heard an argument that the manifest comes from some other OGC documents.

The information contained in the manifest looks more or less ok. But it feels like that part of the spec was written by someone who just wasn't that comfortable with the sql api's that make the rest of the spec, so just wrote from their experience with ows context and atom, and shoved it in to the field. I think geopackage should not depend on an OGC-fluent audience, but an audience of potentially geo-naive implementors. So the manifest should be consistent with the rest of the specification, not consistent with OGC xml.

#### 8.14.1 SWG Response

Thank you for your comment. The manifest table has been removed from the GeoPackage specification. Metadata such as OGC Context documents stored in the

metadata table may be used to provide links to online sources of GeoPackage data. The metadata table metadata column is specified as not null, with mime types other than XML are now supported by a metadata table mime type column.

## 9 Comment: Andreas Matheus 2013-02-08

**PART A**

**1. Evaluator:**

Andreas Matheus, Secure Dimensions GmbH

**2. Submission:**

OGC 12-128r1 Version: 0.2.0 - OpenGIS® GeoPackage Implementation Specification

**PART** B

### 9.1 Comment: Security Considerations

1. Requirement: NA
2. Implementation Specification Section number: NA
3. Implementation Criticality: Minor
4. Comments/justifications for changes

This submission describes a "self-contained, single-file, cross-platform, serverless, transactional, open source RDBMS data container with table definitions, relational integrity constraints, an SQL API exposed via a "C" CLI and JDBC, and an XML manifest that together act as an exchange and direct-use format for multiple types of geospatial data including vector features, individual rasters and tile matrix pyramids, especially on mobile / hand held devices in disconnected or limited network connectivity environments." Because the anticipated use is especially on mobile /handheld devices, an outline about the risks involved seem to be of interest.

As it is good practice to follow other standardization organizations such as OASIS and IETF, I do encourage the submitting organizations to provide security considerations outlining how security regarding confidentiality, integrity, authentication and authorization could be achieved to protect the assets from the GeoPackage.

#### 9.1.1 SWG Response

Thank you for your comment. The following Security Considerations clause has been added to the draft GeoPackage specification:

"Security considerations for implementations utilizing GeoPackages are in the domain of the implementing application, deployment platform, operating system and networking environment. The GeoPackage specification does not place any constraints on application, platform, operating system level or network security."