Testbed-12 GeoPackage Routing and Symbology Engineering Report

Table of Contents

1. Introduction . . . 5
   1.1. Scope . . . 5
   1.2. Document contributor contact points . . . 5
   1.3. Future Work . . . 5
   1.4. Foreword . . . 5
2. References . . . 6
3. Terms and definitions . . . 7
   3.1. Abbreviated terms . . . 7
4. Overview . . . 8
5. Routing Requirements . . . 9
   5.1. Status Quo . . . 9
   5.2. Requirements Statement . . . 9
6. Routing Solutions . . . 10
   6.1. Development Environments . . . 10
   6.2. Routing Algorithms . . . 10
   6.3. Just-in-Time Layer Loading . . . 10
   6.4. Threading . . . 10
   6.5. GeoPackage Import Source . . . 10
   6.6. Use of TINs . . . 11
   6.7. Limitations and Issues . . . 11
      6.7.1. Flexibility of GeoPackage Encoding Standard . . . 11
   6.8. Recommendations . . . 11
      6.8.1. TIN Extension . . . 11
7. Symbology Requirements . . . 12
   7.1. Use Cases . . . 12
   7.2. Status Quo . . . 12
   7.3. Requirements Statement . . . 12
8. Symbology Solutions . . . 13
   8.1. Approaches Considered . . . 13
      8.1.1. Style Encoding . . . 13
   8.2. Recommended Design Decisions and Rationales . . . 16
   8.3. Limitations and Issues . . . 16
Appendix A: Static Feature Symbology Extension . . . 17
   Styles . . . 18
Appendix B: Dynamic Feature Symbology Extension . . . 21
Appendix C: Revision History . . . 25

Publication Date: 2017-05-12

Approval Date: 2017-01-26

Posted Date: 2016-11-16

Reference number of this document: OGC 16-029r1

Reference URL for this document: http://www.opengis.net/doc/PER/t12-A082

Category: Public Engineering Report

Editor: Jeff Yutzler

Title: Testbed-12 GeoPackage Routing and Symbology Engineering Report

---

Testbed-12 GeoPackage Routing and Symbology (16-029r1)

COPYRIGHT

WARNING

destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Abstract

This OGC Engineering Report (ER) describes the results of experiments in OGC Testbed 12 designed to potentially enhance capabilities for symbology and routing [1: Routing is the process of selecting best paths in a network. Routing is performed for many kinds of networks, including the public switched telephone network (circuit switching), electronic data networks (such as the Internet), and transportation networks. This Engineering Report focuses on the latter.] as extensions to the OGC GeoPackage standard. These experiments focused on 1.) methods for providing mounted and/or dismounted (off-road) routing within GeoPackage and 2.) mechanisms for providing user-defined map symbology for features in a GeoPackage structured data store. This ER documents the different approaches considered, design decisions and rationales, limitations, and issues encountered during prototype implementation.

Business Value

This ER describes proposed solutions for filling two interoperability gaps in the OGC baseline. The first is in off-road routing, where there is currently no interoperable means for sharing routes between systems. The second is in user-defined map symbology, where existing approaches such as Symbology Encoding (SE) have not achieved desired objectives of persisting styling rules and making them available to client applications throughout the enterprise.

Technology Value

This ER proposes a way ahead for the the two topic areas. If these approaches prove to be useful, they will be submitted to the GeoPackage SWG for consideration as official extensions to OGC GeoPackage.

How does this ER relate to the work of the Working Group

After producing GeoPackage 1.1, the GeoPackage SWG investigated possible extension areas. This ER presents the SWG with proposed approaches for routing and symbology.

Keywords

ogcdocs, testbed-12, GeoPackage, routing, symbology, styling, cross country, off-road

Proposed OGC Working Group for Review and Approval

GeoPackage SWG

Chapter 1. Introduction

1.1. Scope

This OGC® document describes the results of experiments to perform routing and styling on a GeoPackage. It includes a proposed approach for persisting static and dynamic styling information inside a GeoPackage.

This OGC® document is applicable to the OGC GeoPackage Encoding Standard.

1.2. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Table 1. Contacts

Name	**Organization**
Jeff Yutzler	Image Matters LLC
Chris Clark	Compusult

1.3. Future Work

It is expected that this document may result in new extensions to the OGC GeoPackage Encoding Standard.

1.4. Foreword

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 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.

Chapter 2. References

The following documents are referenced in this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

- OGC 16-037, Testbed-12 GeoPackage US Topo ER
- OGC 06-121r9, OGC® Web Services Common Standard

NOTE: This OWS Common Standard contains a list of normative references that are also applicable to this Implementation Standard.

- OGC 12-128r12, OGC® GeoPackage Encoding Standard 1.1

Chapter 3. Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard [OGC 06-121r9] shall apply.

3.1. Abbreviated terms

- IDE Integrated Development Environment
- iOS an operating system used for mobile devices manufactured by Apple Inc.
- JSON JavaScript Object Notation
- MXD file extension for a map document used by ArcMap
- OS Operating System
- PC Personal Computer
- PCL Portable Class Library
- SC Secure Digital
- SE OGC Symbology Encoding
- SLD OGC styled Layer Descriptor
- SWG Standards Working Group

Chapter 4. Overview

The following table describes the sections that will appear later in this document.

Section Number	Description
5 - Routing Requirements	Description of routing capabilities, status quo, and requirements statement
6 - Routing Solutions	Design decisions needed, alternatives considered, recommendations, and rationales for routing application
7 - Symbology Requirements	Description of symbology capabilities, status quo, and requirements statement
8 - Symbology Solutions	Design decisions needed, alternatives considered, recommendations, and rationales for symbology capabilities
Appendix A - Static Symbology Extension	Proposed GeoPackage extension template for static symbology information
Appendix B - Dynamic Symbology Extension	Proposed GeoPackage extension template for dynamic symbology information
Appendix C	Revision History

Chapter 5. Routing Requirements

The purpose of this mobile application is to provide users the capability to view GeoPackages as a source or layer and create routes (excursions) that performs elevation analysis offline or with no wireless connection availability. Included in the application will be a user preferences activity that will allow the user to select pre-designated weighting rules or cost functions that influence route traversal. These options will be categorized and listed based on traversal method (vehicle (various types), walking/hiking, obstacle avoidance preferences (slope planes, geo-political, and hydrographic elements)). Based on the category selected, a single traversal method method will be selected to determine the appropriate algorithm, or combination of, used for route determination including obstacle avoidance constraints preferences.

5.1. Status Quo

There is currently existing functionality in place to determine routing in the form of C++ libraries. Researchers are unaware of any prior work that provides equivalent capabilities in a form that is readily usable in the target computing environments.

5.2. Requirements Statement

1. There are two target computing environments, PC (using Windows OS) and an tablet/mobile device (using Android OS).
2. The user will have the ability to import a GeoPackage from their local device (i.e. flash, SD card storage) into the application.
3. The user may load one or more GeoPackages into the application, where the GeoPackage is considered "active" and can be used to visualize GeoPackage contents, or may be used computationally to perform analysis.
4. The off-road routing application will apply user-selectable, pre-designated weighting rules to return an acceptable path from an origination point to a destination point.

Chapter 6. Routing Solutions

6.1. Development Environments

We determined that Visual Studio with Xamarin (C#) was the most suitable IDE because of its support for both target computing environments. This approach enabled us to reuse the portable class libraries (PCL) we developed for each platform (Android, Windows, iOS). Any code not specific to an operating system (UI-related) was placed in a PCL therefore minimizing the need to re-write the same functions per OS targeted. This allowed us to develop libraries that could be used for cross-platform (iOS, Android, Windows, etc.) development with limited platform-dependent restrictions (specifically UI restrictions).

We also made the decision to utilize some open-source projects, such as Mapsui and SharpMap. SharpMap is a web-based mapping application that supports GeoPackage usage and Mapsui is a mobile variant of SharpMap, but does not explicitly support GeoPackages. By integrating the two libraries we were able to create a mobile solution with the ability to render GeoPackage tiles.

6.2. Routing Algorithms

We attempted to import or "wrap" existing routing C libraries in a Xamarin portable class library but were unable to do so. Instead we ported the capabilities into a PCL (from C to C#).

6.3. Just-in-Time Layer Loading

Since the size of a GeoPackage can be quite large, we decided to prevent the application from loading a full GeoPackage into device memory. The fact that a GeoPackage can contain multiple tile, feature, and attribute layers, we decided to only reference a layer when needed. Therefore, layer toggling (only loading layer when a user requests it) is only performed on request to minimize memory and processing. Better handling of the renderable layers will need to be discussed in order to leverage memory-management best-practices in the future.

6.4. Threading

At the point a route is requested, we determined that the app would be more efficient in processing if we moved the event onto an asynchronous processing thread. Moving this to an asynchronous thread allowed us to notify the user that route processing is still “working” while it retrieves all the data points and calculates the route for displaying.

6.5. GeoPackage Import Source

Further research and analysis is needed on the GeoPackage import source and procedure. Methods researched include PC transfer (requires external functionality), Bluetooth, website download (requires external functionality), and external SD card. For the purposes of this experiment, we imported a GeoPackage from the device’s flash memory.

6.6. Use of TINs

TINs proved to be a better solution than gridded elevation data in most cases because it provides a place to store a preprocessed cost for routes. Gridded elevation data provides node height which is in itself useful, but makes route processing more cumbersome.

6.7. Limitations and Issues

6.7.1. Flexibility of GeoPackage Encoding Standard

The GeoPackage standard is extraordinarily flexible and, as a result, problematic when developer-defined extensions are created. If instances other than features, tiles, and attributes (mentioned in OGC's Geopackage standard spec document) are used, not all items within the gpkg_contents table would be processed unless they are explicitly programmed for in that particular application. In short, an app that can display/save media information within a GeoPackage may only be able to render data created by that same app, leaving no room for data use outside of that application. This also creates an issue of more than one layer within a GeoPackage not being visible or used to determine route traversal.

6.8. Recommendations

6.8.1. TIN Extension

There is currently no standard for TIN or gridded elevation data within a GeoPackage. In order to be able to process elevation data from a TIN, a conversion process has to take place (from GeoTiff-DEM data to node/edge data file format). This process could only be completed on a former developer's machine given many undocumented libraries of which it contained.

Chapter 7. Symbology Requirements

Using an extension to express a method of styling features tests the extension mechanism and advances the work of the GeoPackage SWG regarding the inclusion of at least one form of styling information in a GeoPackage.

7.1. Use Cases

The SWG has identified 2 use cases which should be addressed in this ER:

- Geometry styling (UC4) - Geometry styles will be applied based on the application of styling directives such as those found in the OGC Style Layer Descriptor (SLD) standard. The output of the interpretation will be a set of symbols/drawing instructions cross-linked to the features they are meant to symbolize. The interpretation rules from the styling will not be carried over.
- Feature Type Styling (UC5) - similar to Geometry Styling, this approach relies on the interpretation of a set of styling directives that process feature attributes to determine a symbolization for a feature. The relevant rules for the style will be stored in the GeoPackage to be interpreted at render time, or be pre-rendered into geometry styling. The benefit of maintaining these rules in the GeoPackage is that the symbolization of a feature can be re-interpreted if its attributes change.

7.2. Status Quo

There is currently no interoperable way of exchanging styling information within a GeoPackage. At best, individual vendors have produced their own approaches to solving the styling requirement.

7.3. Requirements Statement

Testbed 12 generated discussion regarding the implementation of use cases UC4 and UC5. The prototype styling implementation created for the testbed exposed an alternate method of styling within the OGC. The findings may be useful to spur more work developing and extending current styling methodologies or developing new ones.

Chapter 8. Symbology Solutions

Solutions for symbology representation in a GeoPackage are documented in another Engineering Report. As of this writing, this report has a working title of **Testbed-12 GeoPackage US Topo ER**.

In order to achieve the immediate goals for Testbed 12, the following solutions were adopted:

1. Encoding of styles from the USGS topographic style template to be used to render topographic data on a client using a style specification system that employs attribute based symbology assignment.
2. Use of a mobile client to demonstrate offline rendering of vector features and supporting raster data according to use cases
3. Generation of a new GeoPackage extension to support vector feature styles for rendering derived from symbology encoding that supports re-styling after feature changes.

In addition to the symbology encoding solution, there was a need to store the relations between the features and the symbologies in the GeoPackage. In order to do this, a new extension was designed to accommodate this approach. This structure and use of this extension is detailed in the "UML Models for Symbology Extension" of **Testbed-12 GeoPackage US Topo ER**.

8.1. Approaches Considered

Two symbology encoding approaches were considered:

1. OGC Styled Layer Descriptor (SLD) / OGC Symbology Encoding (SE).
2. A JSON encoding developed by Compusult

While the OGC SLD/SE standard was considered, it was more effective to use the JSON approach as it was already proven to work to accomplish the goals of the Testbed, and there are some benefits to using the JSON symbology encoding. Compusult was not aware of any other efforts and decided therefore to use their JSON based in-house approach. This approach is described in the "Symbology Encoding" appendix of **Testbed-12 GeoPackage US Topo ER**.

8.1.1. Style Encoding

Style encoding was accomplished through the use of a stylesheet that defined the appropriate symbology for feature layers based on filter criteria and scale. If there is no filter criteria pertaining to a style, the style was generalized for the whole feature type. In the process of styling the features, ancillary tables for the encoded feature types were created to relate individual features to:

1. A determined symbology based on scale
2. If required, a rule that determined the symbology based on scale and a rule attribute

In the process of constructing the symbology for a feature type, Compusult determined that there are limitations in that the layer hierarchy defined in the Topographic Map Template could not be followed as there was no way of directly encoding the groups implied in the template. There were two ways the feature data could be encoded as closely as possible to the Feature Classes that appear in the GeoDataBase.

Styles covering all required symbologies and feature types defined by definition queries were merged into a set of styles applied to the single feature class. A good example of this is the group of sub layers derived from the GU_Reserve feature class. On their own in the template MXD, there were the following "sub features" each with an implied style based on a criteria.

In this case, the method employed did not segregate the data into separate feature tables in the GeoPackage. Instead, all features remained in the same base table and were simply styled according to the requirements dictated in the original MXD based template.

The other approach was to generate individual feature tables derived from the query definitions in the TNM template. This was the approach chosen as the user would have similar control over each layer as they would when viewing the data in ArcMap.

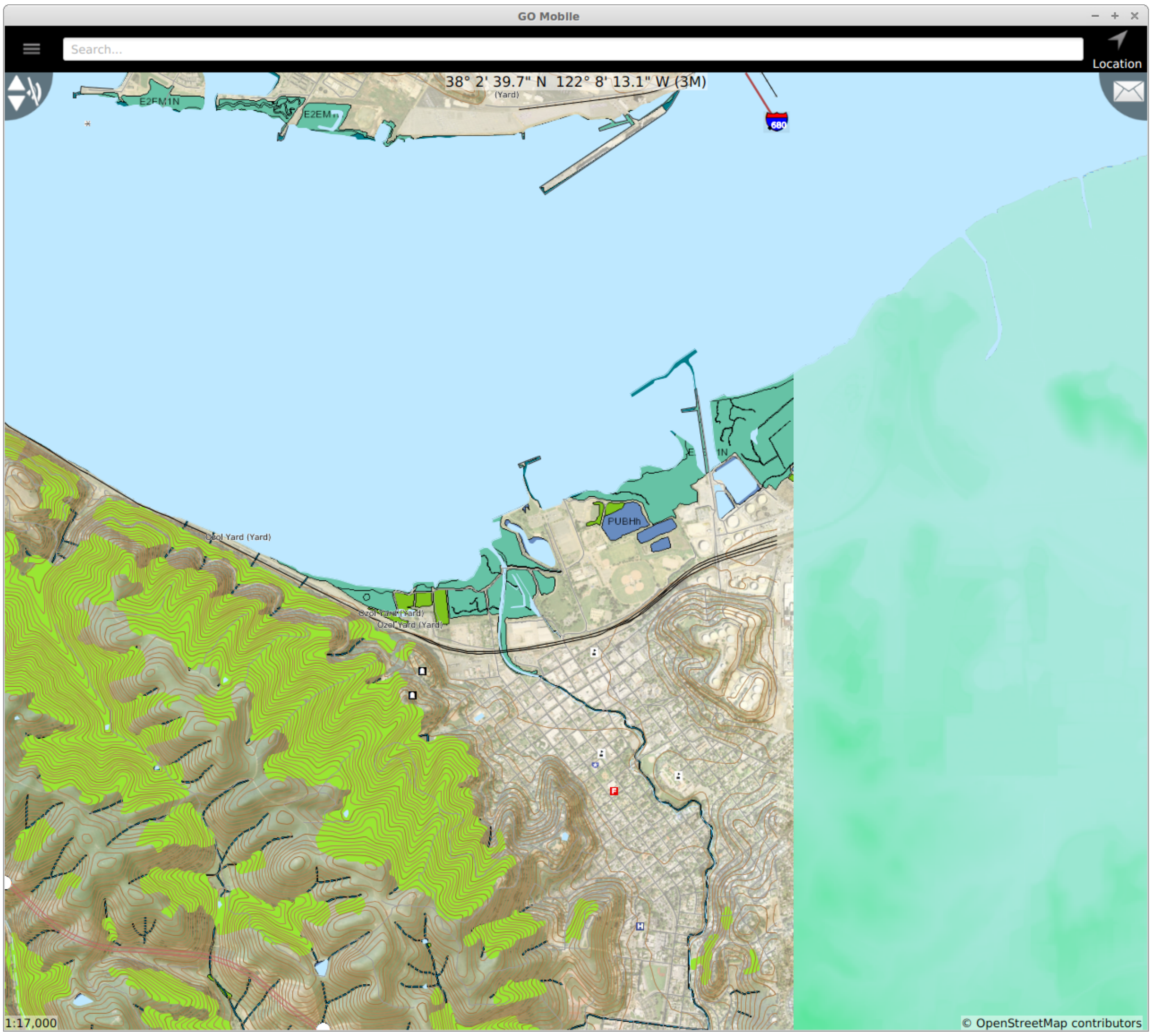
GO Mobile
Search...
Location
38° 2' 39.7" N 122° 8' 13.1" W (3M)
E2FM1N
E2EM
PUBHh
1:17,000
© OpenStreetMap contributors

Figure 1. Styled data for Benicia

This image shows the following USGS feature types superimposed on imagery harvested from the imagery server mentioned above:

Schools,

- Fire Stations,
- Place Names,
- Contours with labels,
- Post Offices
- Rail Line,
- Forested Areas

This image also shows raster imagery from:

- Shaded Relief,
- Imagery,
- Wetlands,
- and 1/9th arc second Elevation Data

This data was derived from the USGS GDB product available at https://prd-tnm.s3.amazonaws.com/StagedProducts/Vector/GDB/VECTOR_3330_Benicia_7_5_Min.zip and generated using the WPS Converter. The GeoPackage is located at https://portal.opengeospatial.org/files/?artifact_id=71497.

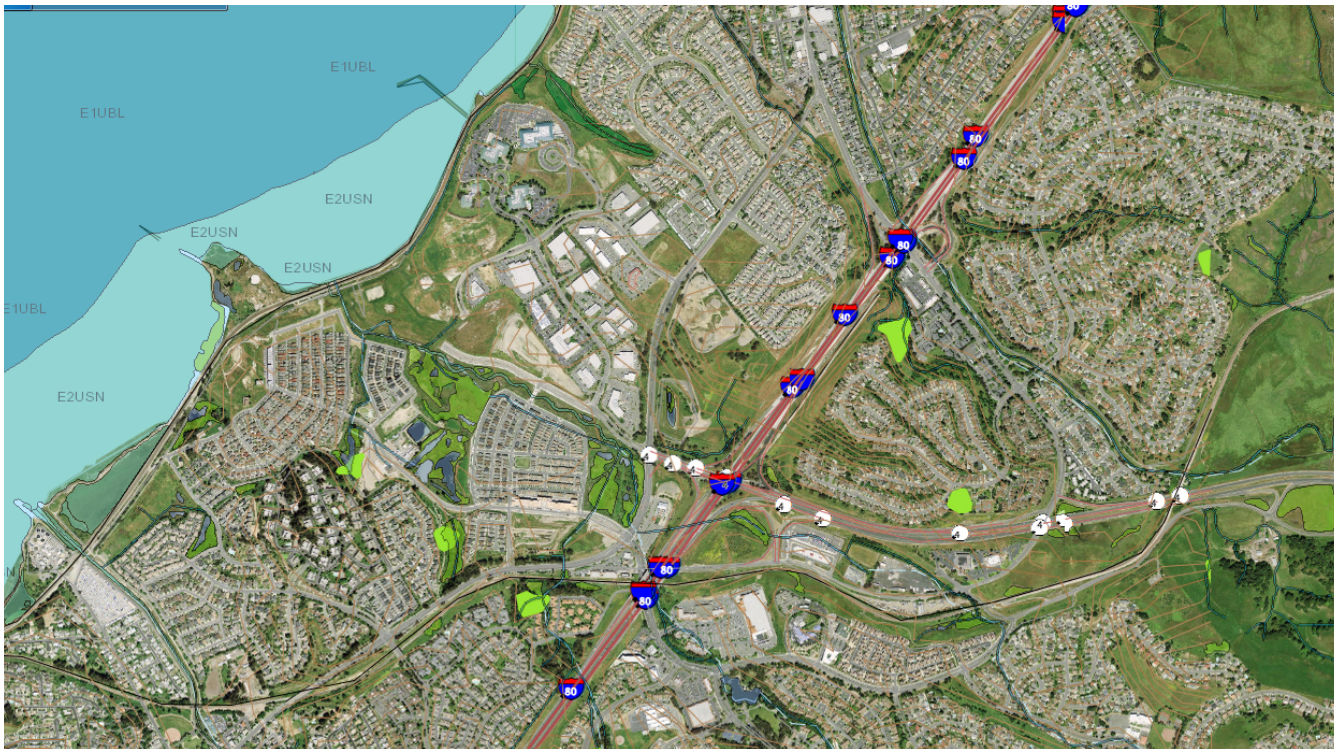

Figure 2. Styled data for Mare Island

This image shows the following USGS feature types superimposed on imagery harvested from the imagery server mentioned above as well as wetlands raster data and shaded relief.

-

8.2. Recommended Design Decisions and Rationales

In order to fulfill the required use cases, the following design approaches are recommended : * Associate direct relations between symbology and features for different scales. This cuts down on interpretation overhead when rendering. This was accomplished by having style tables linked to feature tables using a mapping table between feature id and style id. The mapping approach prevents duplication. * Do not represent symbology instructions using database constructs. It is far better to use a BLOB field to hold the drawing instructions. This supports change and different symbology formats without disrupting the basic structure of the extension. * Do narrow down the symbology formats used. One is probably not enough as different communities have different needs. * Design a JSON encoding for OGC SE (Symbology Encoding) to support both core drawing instructions in addition to supporting vendor specific extensions. * Design an agreed upon attribute based filter language (possibly based directly on SQL). The one used in Testbed 12 is based on SQL and is meant to serve as a "where clause". * Do implement a method of capturing: **painting order of additive styles, and** scales at which the styles are applied.

8.3. Limitations and Issues

While not necessarily a symbology encoding issue, there was a limitation in the approach used in that the presentation layers in the model data used in the experiment is organized as a set of views and not simply feature classes or feature tables. The symbology was applied to these views in the same way as it was applied directly to feature classes. Possible extensions to GeoPackage might include the development of conceptual organizations of data into groups of layers or views of data.

Some sophisticated filter expressions that exist in Esri tools are not supported directly. For example:

```
"T" & Mid([PLSSID]. 5, 3)
```

Ultimately, there are issues regarding how symbology encodings are interpreted. Precise agreement between client implementations on how to interpret and render symbology will prevent the same GeoPackage from looking exactly the same from client to client.

Performance of rendering, especially on less powerful devices requires rendering information for each feature to be kept in the GeoPackage. While this "cache" creates great benefit, it uses extra space in the GeoPackage, beyond the style rules and encodings and the features themselves.

Appendix A: Static Feature Symbology Extension

Introduction

While feature data stored in GeoPackages can be visually represented by client applications in an infinite number of ways, producers of GeoPackage products containing feature data often wish to have these products depicted in a consistent manner based on rules that govern rendering such as color, size, orientation, icons etc.

In order to support this requirement, this extension is provided to store these symbologies and their application to features stored in GeoPackage feature tables.

The extension is designed to facilitate association of individual features with an appropriate symbolization based on viewing scale to determine an appropriate symbology.

The symbolization is "hard-coded" for the feature. Each feature to be rendered should be linked to a symbology created by the producer and stored in the GeoPackage at creation time.

note

This extension was developed for and documented according to its application in the testbed 12 activity and it is not intended to be a complete extension specification for inclusion in the GeoPackage specification without review and revision.

Extension Author

Compusult (cslt)

Extension Template

For each feature table requiring extension for symbology, the following extended tables shall be created and registered in the gpkg_extensions table:

- cslt_<tablename>_style_rules
- cslt_<tablename>_styles
- cslt_<tablename>_style_images
- cslt_<tablename>_style_links

Extension Type

New Requirement Dependent on Clause 2.1.

Applicability

This extension applies to tables defined under Clause 2.1 i.e. vector features.

Scope

This is a read-write extension in that clients may only read from associated symbology to render, or clients may trigger the update of tables that contain the associated style/feature mappings with new symbology.

Requirements

Table Definitions

Styles

```
A GeoPackage that conforms to this extension shall, for each feature table covered by
this extension, contain a table whose name is derived from the feature using the
following template : cslt_<tablename>_styles
```

Table 2. cslt_<tablename>_styles Table

Column Name	Data Type	Description	Key
id	INTEGER	The primary key of the style limited to values in the range [0, 9223372036854775807]	PK
ordinal_pos	INTEGER	indicates the order of application for the style if there are more than one style rules applicable for a feature. Falls into the range [0, 9223372036854775807].	
rule_id	INTEGER	A positive number that is a foreign key to the corresponding rules table	FK
style	TEXT	The encoded style or symbology instructions detailing how the feature should appear	
style_encoding	TEXT	a mime type that indicates how the style is encoded e.g. "application/vnd.ogc.se_xml"	

Column Name	Data Type	Description	Key
priority	REAL	A priority used to weight the symbolization of the feature used to remove cluttering	

Style Rules

```
A GeoPackage that conforms to this extension shall, for each feature table covered by
this extension, contain a table whose name is derived from the feature using the
following template : cslt_<tablename>_style_rules
```

Table 3. cslt_<tablename>_style_rules Table

Column Name	Data Type	Description	Key
id	INTEGER	The primary key of the rule limited to values in the range [0, 9223372036854775807]	PK
ordinal_pos	INTEGER	indicates the order of application for the rule if there are more than one style rules applicable for a feature. Falls into the range [0, 9223372036854775807].	
min_scale_denom	REAL	A real value used for evaluating when the rule should be applied to a feature. When the scale denominator of the visible map is > min_scale_denom, the rule is active	
max_scale_denom	REAL	A real value used for evaluating when the rule should be applied to a feature. When the scale denominator of the visible map is < max_scale_denom, the rule is active	
rule_type	INTEGER	0 means that the style is for a geometry, a 1 means that the style is a label style	

Style Images

```
A GeoPackage that conforms to this extension shall, for each feature table covered by
this extension, contain a table whose name is derived from the feature using the
following template : cslt_<tablename>_style_images.
```

Table 4. cslt_<tablename>_style_images Table

Column Name	Data Type	Description	Key
id	INTEGER	The primary key of the image limited to values in the range [0, 9223372036854775807]	PK
path	TEXT	The original path of the symbol such as "/usr/share/images/tower.png"	
data	BLOB	binary data	

Style Links

```
A GeoPackage that conforms to this extension shall, for each feature table covered by
this extension, contain a table whose name is derived from the feature using the
following template : cslt_<tablename>_style_links.
```

Table 5. cslt_<tablename>_style_links

Column Name	Data Type	Description	Key
feature_id	INTEGER	A foreign key to a feature in the related feature table	FK
rule_id	INTEGER	A foreign key to a rule related rule table	FK

Appendix B: Dynamic Feature Symbology Extension

Introduction

In addition to supporting direct representational styling using static feature symbology extension an additional extension is provided to support dynamic styling of features. Dynamic styling is applicable when the attributes of a feature that determine its appearance have changed. For example, a user changes a road attribute such as lanes from "2" to "4". Styling rules exist that require road features with a lane value of "4" to be rendered with a median. Unless this rule is dynamically interpreted, the road will not be updated visually unless there is a mechanism to find an applicable set of style rules for the changed feature and determine which one applies to the new condition. If such a mechanism exists, the applicable rule will be used to determine the new symbology. In the road case, the expression lanes=4 would be encountered, the corresponding symbology would be set for the feature and clients would now render the feature with a median.

To support this dynamic mechanism, a further extension to the static feature symbology extension is encoded to support the necessary links between symbology and rule expressions.

The core idea of the dynamic symbolization is that a set of expressions are evaluated for any feature in the feature table. If the match expression is satisfied, any style from the dynamic styles table that is linked to that expression is applied to the feature. Application of these styles could occur at run time, or they can be computed and stored in the data model for static styles. The following are detailed descriptions of the extended tables. These extended tables are linked via a mapping in gpkg_extended_contents that maps the feature table to the corresponding style tables below.

Extension Author

Compusult (cslt)

Extension Name or Template

For each feature table requiring symbology, the following extended tables shall be created and registered in the gpkg_extensions table :

- cslt_<tablename>_dynamic_styles
- cslt_<tablename>_dynamic_styles_images
- cslt_<tablename>_dynamic_syles_expressions

Extension Type

New Requirement Dependent on Clause 2.1.

Applicability

This extension applies to tables defined under Clause 2.1 i.e. vector features.

Scope

This is a read-only extension to support clients editing the attributes of features associated with the symbology in such a way as to trigger the update of static style tables that contain the associated style/feature mappings.

Requirements

Table Definitions

```
A GeoPackage that conforms to this extension shall, for each feature table covered by
this extension, contain a table whose name is derived from the feature table name
using the following template : cslt_<tablename>_dynamic_styles
```

The *dynamic_styles table* is used to store feature symbology information and that information's relation to a boolean expression that determines when that rule should be applied to a feature.

An example would be a style for a red road line. A line feature from <tablename> would get its symbology (red line) from this table, additionally, this table would dictate which style would be applied based on a combination of the scale denominator range for that style and a successful evaluation of that style's related boolean expression based on a feature's attributes.

Table 6. cslt_<tablename>_dynamic_styles

Column Name	Data Type	Description	Key
id	INTEGER	The primary key of the rule limited to values in the range [0, 9223372036854775807]	PK
expression_id	INTEGER	A foreign key to the dynamic_style_expressions table for the feature table	FK
min_scale_denom	REAL	A real value used for evaluating when the style should be applied to a feature. When the scale denominator of the visible map is > min_scale_denom, the style is active	

Column Name	Data Type	Description	Key
max_scale_denom	REAL	A real value used for evaluating when the style should be applied to a feature. When the scale denominator of the visible map is < max_scale_denom, the style is active	
style	TEXT	the text encoding of the symbology information for example an OGC symbology encoding document	
style_encoding	TEXT	the mime type of the style encoding such as "application/vnd.ogc.se_xml"	
priority	REAL	indicates the order of application for the styles if there are more than one styles applicable for a feature.	

```
A GeoPackage that conforms to this extension shall, for each feature table covered by
this extension, contain a table whose name is derived from the feature table name
using the following template : cslt_<tablename>_dynamic_style_expressions
```

The _dynamic_style_expressions table holds expressions used to evaluate a feature based on its attributes. If a feature successfully matches an expression, that feature is styled based on the styles that are related to that expression.

Table 7. cslt_<tablename>_dynamic_style_expressions

Column Name	Data Type	Description	Key
id	INTEGER	A foreign key to a feature in the related feature table	PK

Column Name	Data Type	Description	Key
match_expr	TEXT	A SQL expression equivalent to an expression in a "where clause" that can be applied to attributes of the feature, the intention being that it can be used directly in queries against the feature table	
precedence	INTEGER	This value determines the order of evaluation for all match expressions associated with a feature	

```
A GeoPackage that conforms to this extension shall, for each feature table covered by
this extension, contain a table whose name is derived from the feature table name
using the following template : cslt_<tablename>_dynamic_style_images
```

```
Many symbology encodings reference external image files. In order for the image files
to be accessible in a portable and unconnected manner, these images must be stored in
the database and have a mechanism for referencing the images. The
_dynamic_style_images stores the images in data column. The symbology encoding in the
_dynamic_styles table shall use the id in the _dynamic_style_images table to
refer to the image.
```

Table 8. cslt_<tablename>_dynamic_style_images

Column Name	Data Type	Description	Key
id	INTEGER	The primary key of the image limited to values in the range [0, 9223372036854775807]	PK
path	TEXT	The original path of the symbol such as "/usr/share/images/tower.png"	
data	BLOB	binary data	
mime_type	TEXT	mime type of the stored image such as "image/png"	

Appendix C: Revision History

Table 9. Revision History

Date	**Release**	**Editor**	**Primary clauses modified**	**Descriptions**
June 15, 2016	J. Yutzler	.1	all	initial version
October 20, 2016	J. Yutzler	.2	all	comments integrated
November 10, 2016	R. Cass	.3	7, 8	Compusult feedback
November 12, 2016	J. Yutzler	.4	all	comments integrated
November 15, 2016	J. Yutzler	.5	all	light editing