OGC Testbed-13

SWAP Engineering Report
# Table of Contents

1. Summary ........................................................................................................ 4  
   1.1. Requirements ......................................................................................... 4  
   1.2. Prior-After Comparison .......................................................................... 4  
   1.3. Technology Integration Experiments ...................................................... 4  
      1.3.1. Demonstrations .................................................................................. 8  
   1.4. Findings .................................................................................................. 11  
   1.5. Recommendations ................................................................................. 13  
   1.6. What does this ER mean for the Working Group and OGC in general .... 13  
   1.7. Document contributor contact points ..................................................... 13  
   1.8. Future Work .......................................................................................... 13  
   1.9. Foreword .............................................................................................. 13  
2. References ..................................................................................................... 15  
3. Terms and Definitions .................................................................................. 16  
   3.1. Abbreviated Terms ................................................................................ 18  
4. Overview ....................................................................................................... 19  
5. Experiments .................................................................................................. 20  
   5.1. Background ............................................................................................ 20  
      5.1.1. Web Feature Service (WFS) .............................................................. 20  
      5.1.2. Simple Features .............................................................................. 21  
      5.1.3. Google Protocol Buffers (Protobuf) ................................................ 21  
      5.1.4. Apache Avro .................................................................................... 22  
   5.2. Requirements ........................................................................................ 22  
   5.3. Prior-After Comparison ........................................................................ 22  
   5.4. Technology Integration Experiments .................................................... 23  
      5.4.1. Test Suite ........................................................................................ 25  
      5.4.2. Initial Test Results .......................................................................... 29  
      5.5. Demonstration ..................................................................................... 29  
6. Findings and Recommendations .................................................................. 36  
   6.1. Findings ................................................................................................ 36  
   6.2. Recommendations ................................................................................ 38  
Appendix A: Revision History ......................................................................... 39  
Appendix B: Bibliography .............................................................................. 40
OGC Engineering Report

COPYRIGHT

Copyright © 2018 Open Geospatial Consortium. To obtain additional rights of use, visit http://www.opengeospatial.org/

WARNING

This document is not an OGC Standard. This document is an OGC Public Engineering Report created as a deliverable in an OGC Interoperability Initiative and is not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any OGC Engineering Report should not be referenced as required or mandatory technology in procurements. However, the discussions in this document could very well lead to the definition of an OGC Standard.
LICENSE AGREEMENT

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD. THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by 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.
Chapter 1. Summary

This OGC document provides an analysis of the prototype implementations, approaches and performance aspects of data serialization techniques explored in OGC Testbed 13. Specifically, it describes work done during Testbed 13 investigating serialization for geospatial data sets on OGC Web Feature Service (WFS) using Google Protocol Buffers (Protobuf) and Apache Avro.

Protocol buffers are Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data. They are described by Google in the following manner - 'think XML, but smaller, faster, and simpler'. With Protobuf Google indicates developers can define how they want their data to be structured once, then they can use special generated source code to easily write and read structured data to and from a variety of data streams and using a variety of languages. Apache Avro is described as a remote procedure call and data serialization framework developed within Apache's Hadoop project. It uses JavaScript Object Notation (JSON) for defining data types and reportedly serializes data in a compact binary format.

1.1. Requirements

The OGC WFS provides an interoperable method to access and update geodata across network-connected components. However, results from previous OGC activities and operational deployments indicate that transferring large volumes of geodata from a WFS over a network with poor or very low bandwidth can take a significant amount of time, and network capacity.

To help meet this challenge OGC Testbed 13 developed prototype implementations and conducted Technology Integration Experiments to assess optimizing data transfer under bandwidth-constraint conditions. This document discusses geospatial data size reduction techniques, focused on enhancing WFS for serialization using Google Protocol Buffers and Apache Avro.

1.2. Prior-After Comparison

This Testbed 13 work builds on experiments conducted in Testbed 12 investigating compression for geospatial data sets on OGC Web Feature Service (WFS) using W3C Efficient XML Interchange (EXI) Format 1.0 (Second Edition). This document is available at the following link -

http://docs.opengeospatial.org/per/16-055.html

Testbed 13 Technology Integration Experiments used the same test data sets as Testbed 12, allowing comparison of the results.

1.3. Technology Integration Experiments

In OGC Testbed 13 participants investigated serialization techniques for geospatial data sets delivered by WFS Servers and Clients by augmenting WFS with software capable of producing output as a serialized object.

The testing architecture for this part of OGC Testbed 13 was configured using a combination of the following data and components:
• **Vector Data** - Feature data over San Francisco representing points (schools_public_pt.shp), lines (stclines_streets.shp) and polygons (schools_public.shp) formed the test baseline.

• **Serializer/Deserializer** - Component that writes and reads data in the ‘protocol buffers’ serialization format engineered by Google or Avro.

• **SWAP WFS** - WFS augmented with the ability to write vector data into a serialization format.

• **SWAP WFS Client** - Application clients with the ability to request serialized data from a SWAP WFS, using an outputFormat query parameter, with a performance recording module to gather metrics on the size of the resulting serializations.

These components were configured for testing as described in the following sequence diagram:

![Figure 1. SWAP sequence diagram](image)

For compression testing The Carbon Project implemented SWAP WFS, SWAP WFS Clients and Serializer/Deserializer in the following architecture -
The SWAP WFS was based on CarbonCloud WFS, extended with Protobuf capability using protobuf-net. protobuf-net is a contract based serializer for .NET code, that writes data in the ‘protocol buffers’ serialization format engineered by Google. The API follows typical .NET patterns (it is broadly comparable, in usage, to XmlSerializer, DataContractSerializer, etc).

Feature data over San Francisco representing Points (schools_public_pt.shp), Lines (stclines_streets.shp) and MultiPolygon (schools_public.shp) formed the test baseline.

The first step in the development process involved creating objects to serialize structured data. To do this, The Carbon Project developed a description of the data structure needed for Point, Line and MultiPolygon Features. From that, a class can be created that encodes the data with a binary format. The generated class will provide the fields that make up the object and takes care of the details of reading and writing the structure as a unit.

The client was developed to support access during testing and to gather metrics. An example using protobuf is shown below:
Figure 3. Example using protobuf

An example using Avro is shown below:

Figure 4. Example using Avro
Using the performance recording module, information about different compression methods and datasets were developed. Initial test results for WFS with protobuf and Avro serializations are presented in the table below. Uncompressed and GZIP Geography Markup Language (GML) and JSON are shown as well for comparison.

<table>
<thead>
<tr>
<th>Geometry</th>
<th>Format</th>
<th>Compression</th>
<th>Record</th>
<th>Size</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point</td>
<td>GML</td>
<td>None</td>
<td>167</td>
<td>81473</td>
<td>451</td>
</tr>
<tr>
<td>Point</td>
<td>JSON</td>
<td>None</td>
<td>167</td>
<td>55306</td>
<td>142</td>
</tr>
<tr>
<td>Point</td>
<td>GML</td>
<td>Gzip</td>
<td>167</td>
<td>6917</td>
<td>40</td>
</tr>
<tr>
<td>Point</td>
<td>GML</td>
<td>Exi</td>
<td>167</td>
<td>7959</td>
<td>146</td>
</tr>
<tr>
<td>Point</td>
<td>JSON</td>
<td>Gzip</td>
<td>167</td>
<td>6547</td>
<td>439</td>
</tr>
<tr>
<td>Point</td>
<td>Protobuf</td>
<td>None</td>
<td>167</td>
<td>20025</td>
<td>590</td>
</tr>
<tr>
<td>Point</td>
<td>Avro</td>
<td>None</td>
<td>167</td>
<td>18502</td>
<td>562</td>
</tr>
<tr>
<td>Polygon</td>
<td>GML</td>
<td>None</td>
<td>140</td>
<td>232197</td>
<td>97</td>
</tr>
<tr>
<td>Polygon</td>
<td>GML</td>
<td>Gzip</td>
<td>140</td>
<td>58264</td>
<td>89</td>
</tr>
<tr>
<td>Polygon</td>
<td>GML</td>
<td>Exi</td>
<td>140</td>
<td>87078</td>
<td>114</td>
</tr>
<tr>
<td>Polygon</td>
<td>JSON</td>
<td>None</td>
<td>140</td>
<td>204851</td>
<td>116</td>
</tr>
<tr>
<td>Polygon</td>
<td>JSON</td>
<td>Gzip</td>
<td>140</td>
<td>57518</td>
<td>125</td>
</tr>
<tr>
<td>Polygon</td>
<td>Protobuf</td>
<td>None</td>
<td>140</td>
<td>99252</td>
<td>582</td>
</tr>
<tr>
<td>Polygon</td>
<td>Avro</td>
<td>None</td>
<td>140</td>
<td>85540</td>
<td>756</td>
</tr>
<tr>
<td>MultiLineString</td>
<td>GML</td>
<td>None</td>
<td>14971</td>
<td>17331008</td>
<td>4132</td>
</tr>
<tr>
<td>MultiLineString</td>
<td>GML</td>
<td>Gzip</td>
<td>14971</td>
<td>1784649</td>
<td>4449</td>
</tr>
<tr>
<td>MultiLineString</td>
<td>GML</td>
<td>Exi</td>
<td>14971</td>
<td>1699740</td>
<td>7953</td>
</tr>
<tr>
<td>MultiLineString</td>
<td>JSON</td>
<td>None</td>
<td>14971</td>
<td>9509623</td>
<td>5552</td>
</tr>
<tr>
<td>MultiLineString</td>
<td>JSON</td>
<td>Gzip</td>
<td>14971</td>
<td>1629668</td>
<td>5113</td>
</tr>
<tr>
<td>MultiLineString</td>
<td>Protobuf</td>
<td>None</td>
<td>14971</td>
<td>3188829</td>
<td>3587</td>
</tr>
<tr>
<td>MultiLineString</td>
<td>Avro</td>
<td>None</td>
<td>14971</td>
<td>2849578</td>
<td>4308</td>
</tr>
</tbody>
</table>

*Figure 5. Initial test results for WFS with protobuf and Avro serializations*

1.3.1. Demonstrations

Prototype implementations, various approaches, test architectures and performance aspects of geospatial data serialization techniques explored in OGC Testbed 13 were assessed in a simulated disaster response scenario. This scenario, and relevant aspects of serialization WFS, are described in the following graphics.
Earthquake in San Francisco

- Response and recovery operations started – relief resources begin moving in

- Command Center established to coordinate efforts...

*Figure 6. SWAP demo - picture 1*
WFS Serialization

- Humvees get latest geospatial data... serialized so it’s easier to transmit across low bandwidth network

*Figure 7. SWAP demo - picture 2*
1.4. Findings

Technology Integration Experiments conducted in Testbed 13 indicate:

1. It is possible for Protobuf on a WFS to produce an output that is much smaller than GML or JSON.
2. It is possible for Avro on a WFS to produce an output that is much smaller than GML or JSON.
3. In no circumstances was the serialization produced by Protobuf or Avro on a WFS smaller than a GZIP of the same data.
4. Protobuf and Avro on a WFS are able to produce serializations which may be rapidly transmitted to a web client application and rendered dynamically in a map display.
5. To serialize data using Protobuf, a Protobuf ‘object’ must exist. Protobuf objects are specially formed structures tailored to each vector data layer, in the form of a protobuf (or other serialization) object. Protobuf objects are represented in a geospatial software system as a class.
6. At the time this document was prepared a mechanism for creating arbitrary protobuf geospatial objects is not available, since the output end of the serialization has no way of knowing what you are trying to send. Future work may assess runtime-extensible objects for geospatial vector data (rather than those which are fixed at build-time).
Figure 9. Results charts
1.5. Recommendations

Given the results of Technology Integration Experiments conducted in Testbed 13, it may be reasonable to consider advancing a 'Serialization Profile for WFS'. This profile would describe Best Practices for a WFS using serialization techniques for Protobuf and Avro.

This would be beneficial to the OGC community as it may increase the utility of WFS. Protobuf is getting significant uptake in the geospatial community, so such a Best Practice would help align OGC technology with the emerging technology market.

1.6. What does this ER mean for the Working Group and OGC in general

Given the results of Testbed 13 it may be reasonable to suggest Change Requests to WFS for data serialization. Results of the Testbed will be reviewed and an assessment made as to whether significant improvements in transferring large volumes of geodata from a WFS over a network with poor or very low bandwidth are noted.

1.7. Document contributor contact points

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

<table>
<thead>
<tr>
<th>Contacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>Jeff Harrison - Editor</td>
</tr>
<tr>
<td>Mark Mattson</td>
</tr>
</tbody>
</table>

1.8. Future Work

This document was submitted to the OGC WFS Standards Working Group (SWG) for review and comment. It is expected that this document may result in changes in other documents.

Examples of possible future work include:

- Advancing a 'Serialization Profile for WFS' to describe Best Practices for a WFS using serialization techniques for Protobuf and Avro.
- Future testbeds could look into options for returning GZIP-wrapped Geography Markup Language (GML) from WFS.
- Future testbeds could also compare SWAP benefits of GeoPackage against those of GZIP’ed GML, Protobuf, Avro and other formats.

1.9. 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 06-121r9, OGC® Web Services Common Standard

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

- Protobuf Github [https://github.com/google/protobuf]
- Protocol Buffers - Google Developers [https://developers.google.com/protocol-buffers/]
- OGC 09-025r2, OGC® Web Feature Service 2.0 Interface Standard – With Corrigendum, July 2014 [http://docs.opengeospatial.org/is/09-025r2/09-025r2.html]
- OGC 09-026r2, OGC Filter Encoding 2.0 Encoding Standard - With Corrigendum, August 2014 [http://docs.opengeospatial.org/is/09-026r2/09-026r2.html]
- protobuf-net [https://github.com/mgravell/protobuf-net]
- MapBox Vector Tiles [https://www.mapbox.com/vector-tiles/]
- MapBox Vector Tiles Specification [https://www.mapbox.com/vector-tiles/specification/]
- Apache Avro 1.8.2 Documentation [https://avro.apache.org/docs/current/]
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 [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2] shall apply. In addition, the following terms and definitions apply.

- **Apache Avro**
  
  A remote procedure call and data serialization framework developed within Apache's Hadoop project. It uses JSON for defining data types and reportedly serializes data in a compact binary format.

- **Class**
  
  A container for data and code. The data within the class can be accessed with properties. The code is referred to as methods.

- **Client**
  
  The software component that can invoke an operation from a service.

- **Coordinate**
  
  One of a sequence of n numbers designating the position of a point in n-dimensional space.

- **Feature**
  
  An application object that represents a physical entity, e.g. a building, a river, or a person. A feature may or may not have geometric aspects.

- **Filter**
  
  A filter expression predicate expression encoded using XML.

- **GeoJSON**
  
  GeoJSON is an open standard format designed for representing simple geographical features, along with their non-spatial attributes, based on JavaScript Object Notation. The features include points (therefore addresses and locations), line strings (therefore streets, highways and boundaries), polygons (countries, provinces, tracts of land), and multi-part collections of these types.
• Geometry

The geometry data type is used to house information on recognized objects, like points, lines, and polygons.

• Interface

The named set of operations that characterize the behaviour of a service.

• Operation

The specification of a transformation or query that a service may be called to execute.

• .NET Framework

The .NET Framework (pronounced dot net) is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large class library known as Framework Class Library (FCL) and provides language interoperability (each language can use code written in other languages) across several programming languages. Programs written for .NET Framework execute in a software environment (as contrasted to hardware environment) known as Common Language Runtime (CLR), an application virtual machine that provides services such as security, memory management, and exception handling. (As such, computer code written using .NET Framework is called "managed code".) FCL and CLR together constitute .NET Framework.

• Object

An instance of a class in memory.

• Protobuf

Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data.

• Service

The distinct part of the functionality that is provided by an entity through interfaces.

• Simple Features
An Open Geospatial Consortium (OGC) and International Organization for Standardization (ISO) standard (ISO 19125) that specifies a common storage and access model of mostly two-dimensional geometries (point, line, polygon, multi-point, multi-line, etc.) used by geospatial information systems.

- **Simple Feature**

A feature with all geometric attributes described piecewise by straight line or planar interpolation between sets of points

### 3.1. Abbreviated Terms

Some of the more frequently used abbreviated terms in this document include:

- **API** Application Programming Interface
- **COTS** Commercial Off The Shelf
- **DCE** Distributed Computing Environment
- **ER** Engineering Report
- **GML** Geography Markup Language
- **HTML** Hypertext Markup Language
- **HTTP** Hypertext Transfer Protocol
- **ISO** International Organization for Standardization
- **JSON** JavaScript Object Notation
- **OGC** Open Geospatial Consortium
- **RPC** Remote Procedure Call
- **SF** Simple Features
- **SWAP** Size, Weight, and Power
- **TIE** Technology Integration Experiment
- **URL** Uniform Resource Locator
- **W3C** World Wide Web Consortium
- **WWW** World Wide Web
- **WFS** Web Feature Service
- **XSD** XML Schema Definition
Chapter 4. Overview

This OGC document provides an analysis of the prototype implementations, approaches and performance aspects of data size reduction and compression techniques explored in OGC Testbed 13. Specifically, it describes work done during Testbed 13 investigating serialization for geospatial data sets on OGC Web Feature Service (WFS) implementations. The investigation focused on extending WFS with Google Protocol Buffers (Protobuf) and Apache Avro output formats, and the associated performance aspects of data size reduction and serialization techniques.

This document contains the following sections:

- **Preface** - This section presents information on administrative and legal aspects of this Engineering Report (ER).
- **Summary** - This section presents information on scope, what this ER means for the OGC in general and document contributor contact points.
- **References** - This section presents information on documents that are referenced in this Engineering Report.
- **Terms** - This section presents information on terms and abbreviations that are used in this Engineering Report.
- **Experiments** - This section presents information on the component implementations, architecture and the results of Technology Integration Experiments conducted.
- **Findings** - This section summarizes the findings. It also provides forward-looking recommendations.
Chapter 5. Experiments

This OGC document provides an analysis of the prototype implementations, approaches and performance aspects of data serialization techniques explored in OGC Testbed 13. Specifically, it describes work done during Testbed 13 investigating serialization for geospatial data sets on OGC Web Feature Service (WFS) using Google Protocol Buffers (Protobuf) and Apache Avro.

This section provides an analysis of the prototype implementations, various approaches, test architectures and performance aspects of geospatial data serialization techniques explored in OGC Testbed 13 and findings.

This section presents information on:

• Background
• Requirements
• Prior-After Comparison
• Technology Integration Experiments
• Demonstration

5.1. Background

The investigation focused on bringing together four key technologies to assess the associated performance aspects of data size reduction and serialization techniques

• Web Feature Service (WFS)
• Simple Features
• Google Protocol Buffers (Protobuf)
• Apache Avro

5.1.1. Web Feature Service (WFS)

The OGC Web Feature Service (WFS) Implementation Specification allows a client to retrieve geospatial data encoded in Geography Markup Language (GML) and other formats from multiple Web Feature Services. The specification defines operations for data access and manipulation operations on geographic features, using HTTP as the distributed computing platform. Via these interfaces, a Web user or service can combine, use and manage geodata — the feature information behind a map image.

This International Standard specifies the behaviour of a service that provides transactions on and access to geographic features in a manner independent of the underlying data store. It specifies discovery operations, query operations, locking operations, transaction operations and operations to manage stored parameterized query expressions:

• Discovery operations allow the service to be interrogated to determine its capabilities and to retrieve the application schema that defines the feature types that the service offers.
• Query operations allow features or values of feature properties to be retrieved from the underlying data store based upon constraints, defined by the client, on feature properties.

• Locking operations allow exclusive access to features for the purpose of modifying or deleting features.

• Transaction operations allow features to be created, changed, replaced and deleted from the underlying data store.

• Stored query operations allow clients to create, drop, list and described parameterized query expressions that are stored by the server and can be repeatedly invoked using different parameter values.

This International Standard defines eleven operations:

• GetCapabilities (discovery operation)

• DescribeFeatureType (discovery operation)

• GetPropertyValue (query operation)

• GetFeature (query operation)

• GetFeatureWithLock (query & locking operation)

• LockFeature (locking operation)

• Transaction (transaction operation)

• CreateStoredQuery (stored query operation)

• DropStoredQuery (stored query operation)

• ListStoredQueries (stored query operation)

Some WFS servers may also support additional non-GML feature encodings and client applications may access them using the outputFormat parameter domains. However, the WFS International Standard does not describe how a server would operate upon such encodings. This is an important distinction for data serialization interoperability testing, demonstration and operational implementation.

5.1.2. Simple Features

Simple Features is an Open Geospatial Consortium (OGC) and International Organization for Standardization (ISO) standard (ISO 19125) that specifies a common storage and access model of mostly two-dimensional geometries (point, line, polygon, multi-point, multi-line, etc.) used by geospatial information systems.

The Simple Features Implementation Specification for SQL forms the basis of multiple encodings of two-dimensional geometries including GeoJSON, a geospatial data interchange format based on JavaScript Object Notation (JSON).

5.1.3. Google Protocol Buffers (Protobuf)

Protocol buffers are Google's language-neutral, platform-neutral mechanism for serializing structured data. They are described by Google in the following manner - 'think XML, but smaller,
faster, and simpler'. With Protobuf, Google indicates developers can define how they want their
data to be structured once, then they can use special generated source code to write and read
structured data to and from a variety of data streams and using a variety of languages such as Java,
Python, C++, C# and others.

The most widely known implementation of Protobuf in the geospatial community is probably
MapBox Vector Tiles (MVT). According to MapBox documentation, vector tiles "are the vector data
equivalent of image tiles for web mapping, applying the strengths of tiling – developed for caching,
scaling and serving map imagery rapidly – to vector data." In addition, "vector tiles contain vector
data instead of the rendered image. They contain geometries and metadata like road names, place
names, house numbers in a compact, structured format. Vector tiles are rendered only when
requested by a client, like a web browser or a mobile app."

5.1.4. Apache Avro

Apache Avro is a remote procedure call and data serialization framework developed within
Apache’s Hadoop project. It uses JSON for defining data types and reportedly serializes data in a
compact binary format.

Avro provides:

• Rich data structures.

• A compact, fast, binary data format.

• A container file, to store persistent data.

• Remote procedure call (RPC).

• Simple integration with dynamic languages. Code generation is not required to read or write
data files nor to use or implement RPC protocols.

5.2. Requirements

The OGC WFS provides an interoperable method to access and update geodata across network-
connected components. However, results from previous OGC activities and operational
deployments indicate that transferring large volumes of geodata from a WFS over a network with
poor or very low bandwidth can take a significant amount of time, and network capacity.

To help meet this challenge OGC Testbed 13 developed prototype implementations and conducted
Technology Integration Experiments to assess optimizing data transfer under bandwidth-constraint
conditions. The assessment focused on enhancing WFS for serialization using Google Protocol
Buffers and Apache Avro.

5.3. Prior-After Comparison

This Testbed 13 work builds on experiments conducted in Testbed 12 investigating compression for
geospatial data sets on OGC Web Feature Service (WFS) using W3C Efficient XML Interchange (EXI)
Format 1.0 (Second Edition). This document is available at the following link -

http://docs.opengeospatial.org/per/16-055.html
Testbed 13 Technology Integration Experiments used the same test data sets as Testbed 12, allowing comparison of the results.

5.4. Technology Integration Experiments

In OGC Testbed 13 participants investigated serialization techniques for geospatial data sets delivered by WFS Servers and Clients by augmenting WFS with software capable of producing output as a serialized object.

The testing architecture for this part of OGC Testbed 13 was configured using a combination of the following data and components -

- **Vector Data** - Feature data over San Francisco representing points (schools_public_pt.shp), lines (stclines_streets.shp) and polygons (schools_public.shp) formed the test baseline.
- **Serializer/Deserializer** - Component that writes and reads data in the 'protocol buffers' serialization format engineered by Google or Avro.
- **SWAP WFS** - WFS augmented with the ability to write vector data into a serialization format.
- **SWAP WFS Client** - Application clients with the ability to request serialized data from a SWAP WFS, using an outputFormat query parameter, with a performance recording module to gather metrics on the size of the resulting serializations.

These components were configured for testing as described in the following sequence diagram:
Figure 10. SWAP sequence diagram

For compression testing The Carbon Project implemented SWAP WFS, SWAP WFS Clients and Serializer/Deserializer in the following architecture -
5.4.1. Test Suite

The SWAP WFS was based on CarbonCloud WFS, extended with Protobuf capability using protobuf-net. protobuf-net is a contract based serializer for .NET code, that writes data in the 'protocol buffers' serialization format engineered by Google. The API follows typical .NET patterns (it is broadly comparable, in usage, to XmlSerializer, DataContractSerializer, etc).

The following example from protobuf-net documentation shows basic usage of serializer for .NET. Please note protobuf is supported in many programming frameworks.

Step 1 - Decorate your classes

```csharp
// [ProtoContract]
class Person {
    // [ProtoMember(1)]
    public int Id {get;set;}
    // [ProtoMember(2)]
    public string Name {get;set;}
    // [ProtoMember(3)]
    public Address Address {get;set;}
}

// [ProtoContract]
class Address {
    // [ProtoMember(1)]
    public string Line1 {get;set;}
    // [ProtoMember(2)]
    public string Line2 {get;set;}
}
```

Step 2 - Serialize data
This writes a 32 byte file to "person.bin":

```javascript
var person = new Person {
  Id = 12345, Name = "Fred",
  Address = new Address {
    Line1 = "Flat 1",
    Line2 = "The Meadows"
  }
};
using (var file = File.Create("person.bin")) {
  Serializer.Serialize(file, person);
}
```

Step 3 - Deserialize data

This reads the data back from "person.bin":

```csharp
Person newPerson;
using (var file = File.OpenRead("person.bin")) {
  newPerson = Serializer.Deserialize<Person>(file);
}
```

Feature data over San Francisco representing Points (schools_public_pt.shp), Lines (stclines_streets.shp) and MultiPolygon (schools_public.shp) formed the test baseline.

The first step in the development process involved creating objects to serialize structured data. To do this, The Carbon Project developed a description of the data structure needed for Point, Line and MultiPolygon Features. From that, a class can be created that encodes the data with a binary format. The generated class will provide the fields that make up the object and takes care of the details of reading and writing the structure as a unit.

An example is shown below of a protobuf model for the MultiPolygon test data.
The client was developed to support access during testing and to gather metrics. An example using protobuf is shown below:
Figure 12. Example using protobuf

An example using Avro is shown below:

Figure 13. Example using Avro
5.4.2. Initial Test Results

Using the performance recording module, information about different compression methods and datasets were developed. Initial test results for WFS with protobuf and Avro serializations are presented in the table below. Uncompressed and GZIP GML and JSON are shown as well for comparison.

<table>
<thead>
<tr>
<th>Geometry</th>
<th>Format</th>
<th>Compression</th>
<th>Record</th>
<th>Size</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point</td>
<td>GML</td>
<td>None</td>
<td>167</td>
<td>81473</td>
<td>451</td>
</tr>
<tr>
<td>Point</td>
<td>JSON</td>
<td>None</td>
<td>167</td>
<td>55306</td>
<td>142</td>
</tr>
<tr>
<td>Point</td>
<td>GML</td>
<td>Gzip</td>
<td>167</td>
<td>6917</td>
<td>40</td>
</tr>
<tr>
<td>Point</td>
<td>GML</td>
<td>Exi</td>
<td>167</td>
<td>7959</td>
<td>146</td>
</tr>
<tr>
<td>Point</td>
<td>JSON</td>
<td>Gzip</td>
<td>167</td>
<td>6547</td>
<td>439</td>
</tr>
<tr>
<td>Point</td>
<td>Protobuf</td>
<td>None</td>
<td>167</td>
<td>20025</td>
<td>590</td>
</tr>
<tr>
<td>Point</td>
<td>Avro</td>
<td>None</td>
<td>167</td>
<td>18502</td>
<td>562</td>
</tr>
<tr>
<td>Polygon</td>
<td>GML</td>
<td>None</td>
<td>140</td>
<td>232197</td>
<td>97</td>
</tr>
<tr>
<td>Polygon</td>
<td>GML</td>
<td>Gzip</td>
<td>140</td>
<td>58264</td>
<td>89</td>
</tr>
<tr>
<td>Polygon</td>
<td>GML</td>
<td>Exi</td>
<td>140</td>
<td>87078</td>
<td>114</td>
</tr>
<tr>
<td>Polygon</td>
<td>JSON</td>
<td>None</td>
<td>140</td>
<td>204851</td>
<td>116</td>
</tr>
<tr>
<td>Polygon</td>
<td>JSON</td>
<td>Gzip</td>
<td>140</td>
<td>57518</td>
<td>125</td>
</tr>
<tr>
<td>Polygon</td>
<td>Protobuf</td>
<td>None</td>
<td>140</td>
<td>99252</td>
<td>582</td>
</tr>
<tr>
<td>Polygon</td>
<td>Avro</td>
<td>None</td>
<td>140</td>
<td>85540</td>
<td>756</td>
</tr>
<tr>
<td>MultiLineString</td>
<td>GML</td>
<td>None</td>
<td>14971</td>
<td>17331008</td>
<td>4132</td>
</tr>
<tr>
<td>MultiLineString</td>
<td>GML</td>
<td>Gzip</td>
<td>14971</td>
<td>1784649</td>
<td>4449</td>
</tr>
<tr>
<td>MultiLineString</td>
<td>GML</td>
<td>Exi</td>
<td>14971</td>
<td>1699740</td>
<td>7953</td>
</tr>
<tr>
<td>MultiLineString</td>
<td>JSON</td>
<td>None</td>
<td>14971</td>
<td>9509623</td>
<td>5552</td>
</tr>
<tr>
<td>MultiLineString</td>
<td>JSON</td>
<td>Gzip</td>
<td>14971</td>
<td>1629668</td>
<td>5113</td>
</tr>
<tr>
<td>MultiLineString</td>
<td>Protobuf</td>
<td>None</td>
<td>14971</td>
<td>3188829</td>
<td>3587</td>
</tr>
<tr>
<td>MultiLineString</td>
<td>Avro</td>
<td>None</td>
<td>14971</td>
<td>2849578</td>
<td>4308</td>
</tr>
</tbody>
</table>

*Figure 14. Initial test results for WFS with protobuf and Avro serializations*

Findings are discussed in the 'Findings' section of this Engineering Report.

5.5. Demonstration

Prototype implementations, various approaches, test architectures and performance aspects of geospatial data serialization techniques explored in OGC Testbed 13 were assessed in a simulated disaster response scenario. This scenario, and relevant aspects of serialization WFS, are described in the following graphics.
Earthquake in San Francisco

- Response and recovery operations started – relief resources begin moving in

- Command Center established to coordinate efforts...

Figure 15. SWAP demo - picture 1
Earthquake in San Francisco

- Updates Common Operating Picture needed throughout day

- Geospatial data needs to be sent from Command Center to multiple Humvees

- \textit{Bandwidth limited}

\textit{Figure 16. SWAP demo - picture 2}
WFS Serialization

- Lightweight OGC Web Feature Service (WFS) in Command Center

- Humvees have mapping systems and are deployed throughout area

- Low-bandwidth emergency networks available...

*Figure 17. SWAP demo - picture 3*
WFS Serialization

- Humvee sends *GetFeature Request* to the Command Center WFS ...

*Figure 18. SWAP demo - picture 4*
WFS Serialization

- Humvees get latest geospatial data... serialized so it’s easier to transmit across low bandwidth network

*Figure 19. SWAP demo - picture 5*
WFS Serialization

- Command Center uses WFS Serialization tools to select best methods, such as Protobuf or Avro to transmit the updates...

- Carbon Project WFS...

*Figure 20. SWAP demo - picture 6*
Chapter 6. Findings and Recommendations

OGC Testbed 13 investigated approaches and performance aspects of data serialization techniques. Specifically, it assessed serialization for geospatial data sets on OGC Web Feature Service (WFS) using Google Protocol Buffers (Protobuf) and Apache Avro.

6.1. Findings

Technology Integration Experiments conducted in Testbed 13 indicate:

1. It is possible for Protobuf on a WFS to produce an output that is much smaller than GML or JSON.

2. It is possible for Avro on a WFS to produce an output that is much smaller than GML or JSON.

3. In no circumstances was the serialization produced by Protobuf or Avro on a WFS smaller than a GZIP of the same data.

4. Protobuf and Avro on a WFS are able to produce serializations which may be rapidly transmitted to a web client application and rendered dynamically in a map display.

5. To serialize data using Protobuf, a Protobuf ‘object’ must exist. Protobuf objects are specially formed structures tailored to each vector data layer, in the form of a protobuf (or other serialization) object. Protobuf objects are represented in a geospatial software system as a class.

6. At the time this document was prepared a mechanism for creating arbitrary protobuf geospatial objects is not available, since the output end of the serialization has no way of knowing what you are trying to send. Future work may assess runtime-extensible objects for geospatial vector data (rather than those which are fixed at build-time)
6.2. Recommendations

Given the results of Technology Integration Experiments conducted in Testbed 13 it may be reasonable to consider advancing a 'Serialization Profile for WFS'. This profile would describe Best Practices for a WFS using serialization techniques for Protobuf and Avro.

This would be beneficial to the OGC community as it may increase the utility of WFS. Protobuf is getting significant uptake in the geospatial community, so such a Best Practice would help align OGC technology with the emerging technology market.
# Appendix A: Revision History

**Revision History**

<table>
<thead>
<tr>
<th>Date</th>
<th>Release</th>
<th>Editor</th>
<th>Primary clauses modified</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 15, 2017</td>
<td>Initial Engineering Reports (IER)</td>
<td>J. Harrison</td>
<td>all</td>
<td>initial version</td>
</tr>
<tr>
<td>July 31, 2017</td>
<td>Updated Content</td>
<td>J. Harrison</td>
<td>all</td>
<td>architecture and content integrated</td>
</tr>
<tr>
<td>September 30, 2017</td>
<td>Preliminary Draft Engineering Reports (DERs)</td>
<td>J. Harrison</td>
<td>all</td>
<td>added TIE content and preparation for publication</td>
</tr>
</tbody>
</table>
Appendix B: Bibliography

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 06-121r9, OGC® Web Services Common Standard

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


