Open Geospatial Consortium

Approval Date: 2013-01-18

Posted Date: 2012-12-07

Publication Date: 2013-03-26

Reference number of this document: OGC 12-097

OGC URI: http://www.opengis.net/def/doc-type/per/OWS-9-bulk-data-transfer-gml

Category: Public Engineering Report

Editor(s): Jeff Harrison

OGC® OWS-9 Engineering Report - SSI - Bulk Data Transfer (GML Streaming)

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.

Document type:	OGC® Engineering Report
Document subtype:	NA
Document stage:	Approved for public release
Document language:	English

Abstract

This document provides a description of the Bulk Data Transfer investigations related to Geography Markup Language (GML) streaming and feature data transportation implemented in the OGC OWS-9 test bed.

This document extends the concept of Bulk Data Transfer to the dissemination of large payloads consisting of geospatial data sets and/or collections of data sets between machines that are connected via a network.

This document also describes the delivery of large payloads consisting of geospatial data sets and/or collections of data sets to SpatiaLite/SQLite to store the data for use by mobile applications.

Keywords

ogcdoc, ows9, ows-9, bulk data transfer, gml

What is OGC Web Services 9 (OWS-9)?

OWS-9 builds on the outcomes of prior OGC interoperability initiatives and is organized around the following threads:

- **Aviation**: Develop and demonstrate the use of the Aeronautical Information Exchange Model (AIXM) and the Weather Exchange Model (WXXM) in an OGC Web Services environment, focusing on support for several Single European Sky ATM Research (SESAR) project requirements as well as FAA (US Federal Aviation Administration) Aeronautical Information Management (AIM) and Aircraft Access to SWIM (System Wide Information Management) (AAtS) requirements.

- **Cross-Community Interoperability (CCI)**: Build on the CCI work accomplished in OWS–8 by increasing interoperability within communities sharing geospatial data, focusing on semantic mediation, query results delivery, data provenance and quality and Single Point of Entry Global Gazetteer.

- **Security and Services Interoperability (SSI)**: Investigate 5 main activities: Security Management, OGC Geography Markup Language (GML) Encoding Standard Application Schema UGAS (UML to GML Application Schema) Updates, Web Services Façade, Reference Architecture Profiling, and Bulk Data Transfer.

- **OWS Innovations**: Explore topics that represent either new areas of work for the Consortium (such as GPS and Mobile Applications), a desire for new approaches to

existing technologies to solve new challenges (such as the OGC Web Coverage Service (WCS) work), or some combination of the two.

- **Compliance & Interoperability Testing & Evaluation (CITE)**: Develop a suite of compliance test scripts for testing and validation of products with interfaces implementing the following OGC standards: Web Map Service (WMS) 1.3 Interface Standard, Web Feature Service (WFS) 2.0 Interface Standard, Geography Markup Language (GML) 3.2.1 Encoding Standard, OWS Context 1.0 (candidate encoding standard), Sensor Web Enablement (SWE) standards, Web Coverage Service for Earth Observation (WCS-EO) 1.0 Interface Standard, and TEAM (Test, Evaluation, And Measurement) Engine Capabilities.

The OWS-9 sponsors are: AGC (Army Geospatial Center, US Army Corps of Engineers), CREAF-GeoViQua-EC, EUROCONTROL, FAA (US Federal Aviation Administration), GeoConnections - Natural Resources Canada, Lockheed Martin Corporation, NASA (US National Aeronautics and Space Administration), NGA (US National Geospatial-Intelligence Agency), USGS (US Geological Survey), UK DSTL (UK MoD Defence Science and Technology Laboratory).

License Agreement

Contents

Page

1 Introduction 7
1.1 Scope 7
1.2 Document contributor contact points 8
1.3 Revision history 8
1.4 Future work 8
1.5 Forward 9
2 References 9
3 Terms and definitions 9
4 Conventions 11
4.1 Abbreviated terms 11
4.2 UML notation 12
5 Bulk Data Transfer using Streaming 12
6 Basic Workflow for Streaming GML 13
7 Testing the Basic Workflow for Streaming GML 14
7.1 Service Description Handling 16
7.2 Data Access Testing 16
7.3 Testing Access with OGC Mobile Client Applications 20
7.4 Transaction Handling in the Basic Workflow for Streaming GML 21
7.5 Other Client Applications 21
8 SQLite/SpatiaLite Workflow for GML Streaming 22
8.1 SQLite/SpatiaLite and GeoPackage Overview 23
8.2 Creating GeoPackage from WFS 24
8.3 Using GeoPackage for Connected/Disconnected Mobile Apps 27
9 Recommendations 29

Figures

Page

Figure 1 – Basic operation sequence for GML retrieval using streaming from WFS 14

Figure 2 – OSM test data over Haiti as GML in Gaia 16

Figure 3 - GML streaming to the Gaia application during testing..... 18

Figure 4 - GML Streaming to a Field User on The Carbon Project Android app 20

Figure 5 - GML Streaming accessed by Lockheed Martin application 22

Figure 6 - Creating a GeoPackage from Streaming GML 25

Figure 7 - GeoPackage mobile app on Android..... 28

Figure 8 – Sequence Diagram for GeoPackage creation from Streaming GML 29

OGC® OWS-9 Engineering Report - SSI - Bulk Data Transfer (GML Streaming)

1 Introduction

1.1 Scope

This document provides a description of the Bulk Data Transfer investigations related to Geography Markup Language (GML) streaming and feature data transportation implemented in the OGC OWS-9 test bed.

This document builds on work conducted in the OGC OWS-8 test bed, where methods for distributing geospatial data sets and/or collections of data sets between machines that may or may not be connected via a network were developed. This approach defined a container format that is ZIP compressed and contains feature data, feature schema and topology encoded using GML and metadata encoded using ISO19115.

This document extends the concept of Bulk Data Transfer to the dissemination of large payloads consisting of geospatial data sets and/or collections of data sets between machines that are connected via a network. Specifically, the approach investigated is designed for scenarios where a geospatial data producer wants to write large amounts of GML data directly (either synchronously or asynchronously) to a stream which can be read by a client application. Streaming services described in this document currently implement the OGC Web Feature Service (WFS) standard with digital data payload format as GML.

Geospatial data sets and/or collections of geospatial data sets described in this document are geographic features. A feature is an abstraction of real world phenomena. A geographic feature is a feature associated with a location relative to the Earth, and a digital representation of the real world encoded as a GML document.

This document also describes the delivery of large payloads consisting of geospatial data sets and/or collections of data sets to SpatiaLite to store the data for use by mobile applications. SpatiaLite is an open source library that extends the SQLite core to support

Spatial SQL capabilities. SQLite is a software library that implements a self-contained, serverless, transactional SQL database engine. SpatiaLite implementations described in this document implement the draft GeoPackage specification. GeoPackage is an open, non-proprietary, platform-independent GeoPackage container for distribution and direct use of all kinds of geospatial data.

1.2 Document contributor contact points

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

Name	Organization
Jeff Harrison	The Carbon Project
Mark Mattson	The Carbon Project
Adam Lloyd	Lockheed Martin

1.3 Revision history

Date	Release	Editor	Primary clauses modified	Description
2012-12-05	0.0.1	Harrison	All	New draft document
2012-12-07	0.0.5	Harrison	All	Draft document was edited throughout with contributions from Messrs. Mattson and Lloyd

1.4 Future work

Improvements in this document are desirable to assess additional methods to disseminate large geospatial data sets and/or collections of data sets between machines connected/disconnected via a network. Specifically, the Streaming GML approach should be tested with OGC Web Feature Services (WFS) delivering Geography Markup Language (GML; ISO 19136) enhanced with “Resolution Parameters” on the GetFeature Request to support data access client applications using density-specific content models. In addition, Transactions into GeoPackage and GeoSynchronization with Streaming GML and GeoPackages should be assessed further.

API enhancements to provide low latency access to real-time streams of rapidly updated feature data may also be investigated. Such implementations may include a streaming client able to access pushed messages without polling a service endpoint. In addition, the use of W3C Server-Sent Events may be investigated.

1.5 Forward

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.

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 09-025r1, OpenGIS Web Feature Service 2.0 Interface Standard (also ISO 19142)

OGC 09-026r1, OpenGIS Filter Encoding 2.0 Encoding Standard

OGC 10-100r3, Geography Markup Language (GML) simple features profile

OGC 06-121r3, OGC® Web Services Common Standard

OGC 12-128r1 GeoPackage Implementation Specification (Draft)

3 Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard [OGC 06-121r3] shall apply. In addition, the following terms and definitions apply:

Attribute - ⟨XML⟩ name-value pair contained in an element
[ISO 19136:2007, definition 4.1.3]

Client application - software component that can invoke an **operation** from a **server**
[ISO 19128:2005, definition 4.1]

Coordinate - one of a sequence of *n* numbers designating the position of a point in *n*-dimensional space
[ISO 19111:2007, definition 4.5]

Coordinate reference system - **coordinate system** that is related to an object by a datum
[ISO 19111:2007, definition 4.8]

Coordinate system - set of mathematical rules for specifying how **coordinates** are to be assigned to points
[ISO 19111:2007, definition 4.10]

Element - ⟨XML⟩ basic information item of an XML document containing child elements, **attributes** and character data
[ISO 19136:2007, definition 4.1.23]

Feature - abstraction of real world phenomena
[ISO 19101:2002, definition 4.11]

NOTE: A feature can occur as a type or an instance. It is intended that the term “feature type” or “feature instance” be used when only one is meant.

Filter capabilities XML - metadata, encoded in XML that describes which **predicates** defined in this International Standard a system implements

Filter expression - predicate expression encoded using XML

Interface - named set of **operations** that characterize the behavior of an entity
[ISO 19119:2005, definition 4.2]

Namespace - ⟨XML⟩ collection of names, identified by a URI reference which are used in XML documents as **element** names and **attribute** names
[W3C XML Namespaces]

Operation - specification of a transformation or query that an object may be called to execute
[ISO 19119:2005, definition 4.3]

Property - facet or **attribute** of an object referenced by a name

Request - invocation of an **operation** by a **client**
[ISO 19128:2005, definition 4.10]

Response - result of an **operation** returned from a **server** to a **client**
[ISO 19128:2005, definition 4.11]

Service - distinct part of the functionality that is provided by an entity through **interfaces**
[ISO 19119:2005, definition 4.1]

Server - particular instance of a **service**
[ISO 19128:2005, definition 4.12]

Uniform Resource Identifier URI - unique identifier for a **resource**, structured in conformance with IETF RFC 2396
[ISO 19136:2007, definition 4.1.65]

4 Conventions

4.1 Abbreviated terms

API	Application Program Interface
COTS	Commercial off The Shelf
CRS	Coordinate Reference System
ER	Engineering Report
FES	Filter Encoding Standard
GML	Geography Markup Language
GMLSF	Simple Feature Geography Markup Language
HTTP	Hypertext Transfer Protocol
KVP	Keyword-Value Pair
MIME	Multipurpose Internet Mail Extensions
OSM	OpenStreetMap

TDS	Topographic Data Store
UML	Unified Modeling Language
WFS	Web Feature Service
XML	Extensible Markup Language

4.2 UML notation

Most diagrams that appear in this standard are presented using the Unified Modeling Language (UML) static structure diagram, as described in Subclause 5.2 of [OGC 06-121r3].

5 Bulk Data Transfer using Streaming

OGC Web Services (OWS), including GML services such as the OGC Web Feature Service, define interfaces for data access and update operations using HTTP as the distributed computing platform. Clients interact with OWS by submitting Requests and getting Responses from Services. However, underlying this approach there may be limitations for Services and Clients. As an example, a WFS implementation may construct FeatureCollection responses by buffering them into memory, and the size will depend on the system configuration and memory allocated. In other words, there may be a limit on the size of returned data since the entire content of a message is present in memory before it is sent. While this is a good strategy for many scenarios, large messages, such as GML data set for an entire country, could exhaust a system's resources.

In OWS-9 integration and testing strategy to deal with such large payloads was streaming. This strategy acknowledges that while messages, especially those expressed in XML (GML), are traditionally thought of as being relatively compact data packages, a message might be multiple gigabytes in size and resemble a continuous data stream more than a data package. In fact, implementers of services that deliver GML have seen this behavior many times in ‘real world’ work implementing OGC-based systems. Therefore this thread of OWS-9 investigated methods for requesting data transferred in a ‘streaming mode’ instead of a ‘buffered mode’.

In such a streaming mode, a Data Provider (the data Service) makes the contents of the message body available to a Data Client (the data requester) in the form of a stream, and the message infrastructure continuously forwards the data from sender to receiver as it becomes available.

6 Basic Workflow for Streaming GML

The streaming approach may be used in WFS serving a set of features encoded as GML. As part of this project a cloud-based WFS was established by The Carbon Project to support testing this 'FeatureSource' workflow. Figure 1 and the following descriptions illustrate the sequence of operation calls to obtain GML as streamed data from a datastore across the WFS interface.

1. A client application would request a capabilities document from the WFS supporting streaming. Such a document contains a description of the operations that the WFS supports and a list of the feature types that the service offers.
2. A client application (optionally) makes a request to a web feature service for the definition of one or more of the feature types that the WFS supporting streaming can service using the DescribeFeatureType operation.
3. Based on the definition of the FeatureType the client application generates a request as specified in the OGC WFS interface. The WFS is invoked to read and service the request
4. A response document containing zero or more features is generated by writing directly (either synchronously or asynchronously) to a stream on the server implementation software. The stream is read by a client application.

The streaming approach used in OWS-9 for Bulk Data Transfer was implemented as a server-side design choice. This approach has the advantage of increasing the performance of GML delivery while requiring no modifications to existing client applications since the required functionality is 'wrapped' in a WFS interface.. This design choice was selected to provide lower the bar for implementation of higher performance architectures to deliver large amounts of GML to requesting client applications.

To enable this approach the PushStreamContent class in .NET 4.5, which works with the new Web API stack, was used. In this approach streamed transfers expose the message as a stream. The client starts processing the message before it is completely delivered. As such, streamed transfers can improve the scalability of a service by eliminating the requirement for large memory buffers. It should be noted that whether using streamed transfer mode improves scalability depends on the size of the messages being transferred.

In this workflow the BBOX Spatial Operator was calculated as each record is read in. This was done because the BBOX isn't known at first – so as the features get streamed out, the BBOX gets built. The result of this approach is that the BBOX appears after the features in the result set. Typically, it is before the features in a GML document.

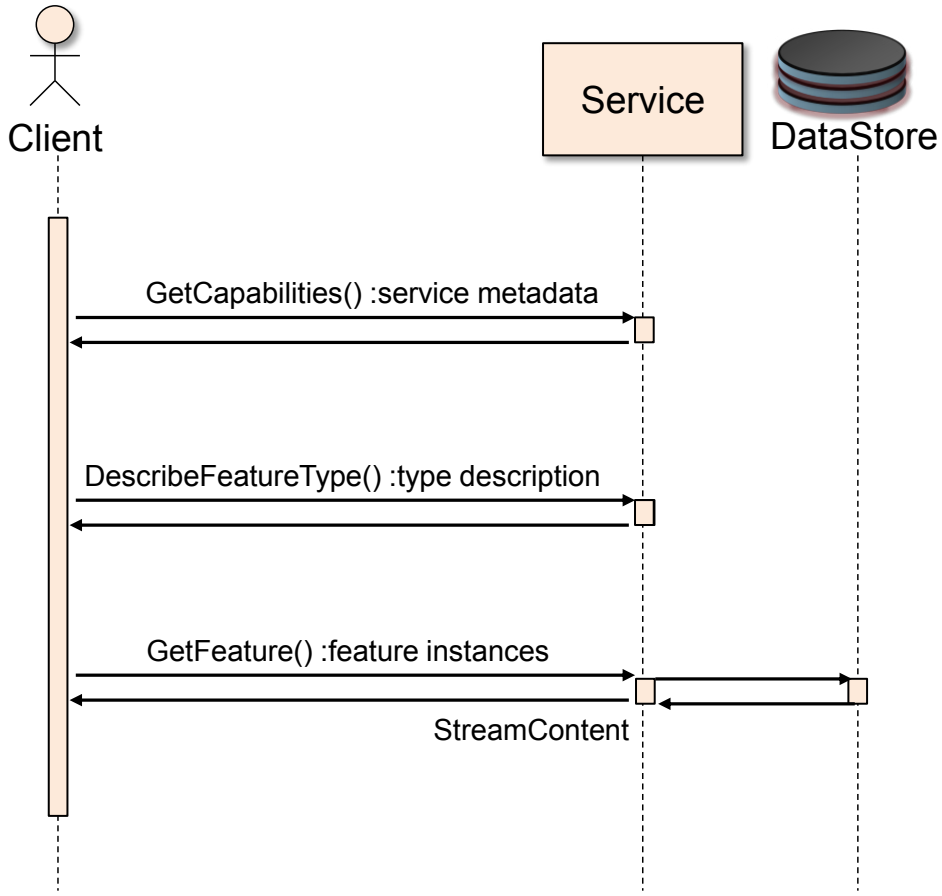

Figure 1 – Basic operation sequence for GML retrieval using streaming from WFS

The streaming approach used in OWS-9 for Bulk Data Transfer was tested in multiple client application configurations as described in the following sections.

7 Testing the Basic Workflow for Streaming GML

To assess the utility of the streaming approach in WFS with features encoded as GML, the project established a test data service with feature data sets over Haiti and the

Dominican Republic on the Microsoft Azure Cloud. OpenStreetMap data representing the following features was obtained as Shapefiles, FeatureTypes were created and data was imported into the test service on the Microsoft Azure Cloud:

Table 1 – OpenStreetMap test data for GML Streaming

FeatureType Name	GML Geometry	Original Data Size
Points	Point	2 MB
Roads	MultiLineString	33 MB
Buildings	MultiPolygon	25 MB

Additional data sets such as Landuse, Railways, Powerlines, Electric Power infrastructure were also obtained from OpenStreetMap to serve as context. The data was accessed as Streaming GML in The Carbon Project's Gaia application and rendered as a map display to support qualitative assessment and OWS-9 demonstration scenarios (see Figure 2).

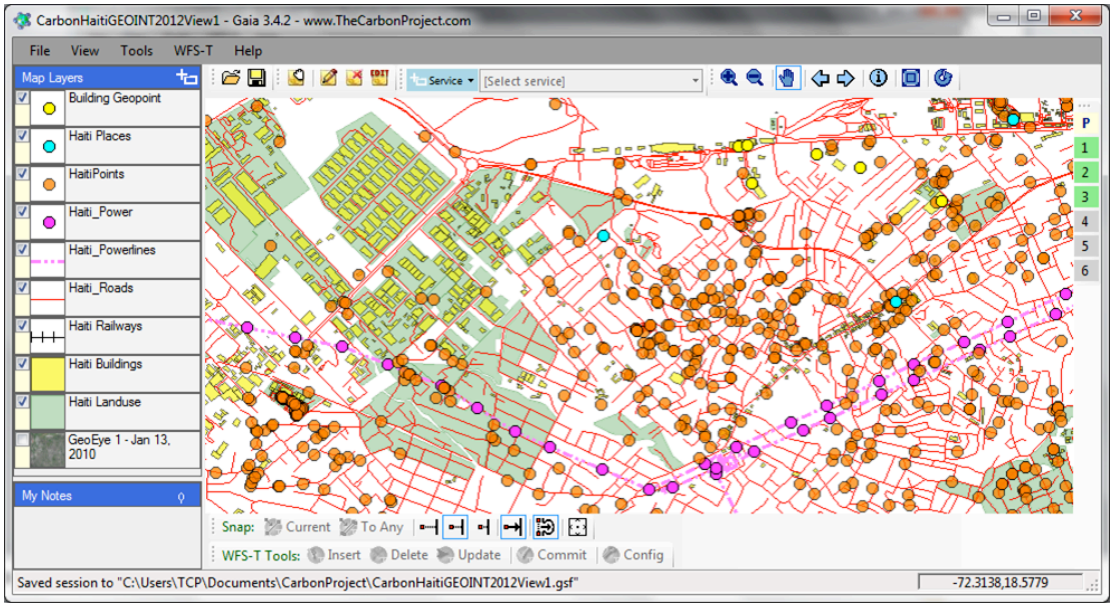

Figure 2 – OSM test data over Haiti as GML in Gaia

Haiti and the Dominican Republic were selected for testing due to the requirements of OWS-9 scenarios. OSM data is available for most of the world and other areas in the United States and Africa have also been loaded on the streaming GML services. The Streaming GML approach also supports other feature data models including, for example, the Topographic Data Store (TDS) Content Specification.

7.1 Service Description Handling

Once deployed, service description as Capabilities and DescribeFeature operations were tested.

7.2 Data Access Testing

Feature access using GetFeature operations were tested with Gaia, mobile applications on Android and other applications developed using ESRI Flex.

Initial testing was conducted using The Carbon Project's Gaia application to generate test requests. A sample query text for a feature request on GML Point features is provided below:

```
<?xml version="1.0" encoding="utf-8"?><GetFeature
xmlns="http://www.opengis.net/wfs"
xmlns:cp="http://thecarbonproject.com"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:gml="http://www.opengis.net/gml" service="WFS" version="1.1.0"
outputFormat="text/xml; subtype=gml/3.1.1" maxFeatures="50000"
handle="" ><Query typeName="cp:HaitiPoints" srsName="EPSG:4326"
><ogc:Filter><ogc:Within><ogc:PropertyName>shape</ogc:PropertyName><gml
:Envelope srsName="EPSG:4326"
xmlns:gml="http://www.opengis.net/gml"><gml:lowerCorner>-
72.3692407366058 18.5340953146098</gml:lowerCorner><gml:upperCorner>-
72.2358722981114
18.5966590713815</gml:upperCorner></gml:Envelope></ogc:Within></ogc:Fil
ter></Query></GetFeature>
```

A sample query text for a feature request on GML MultiLineString features is provided below:

```
<?xml version="1.0" encoding="utf-8"?><GetFeature
xmlns="http://www.opengis.net/wfs"
xmlns:cp="http://thecarbonproject.com"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:gml="http://www.opengis.net/gml" service="WFS" version="1.1.0"
outputFormat="text/xml; subtype=gml/3.1.1" maxFeatures="50000"
handle="" ><Query typeName="cp:Haiti_Roads" srsName="EPSG:4326"
><ogc:Filter><ogc:Within><ogc:PropertyName>shape</ogc:PropertyName><gml
:Envelope srsName="EPSG:4326"
xmlns:gml="http://www.opengis.net/gml"><gml:lowerCorner>-
72.3692407366058 18.5340953146098</gml:lowerCorner><gml:upperCorner>-
72.2358722981114
18.5966590713815</gml:upperCorner></gml:Envelope></ogc:Within></ogc:Fil
ter></Query></GetFeature>
```

A sample query text for a feature request on GML MultiPolygon features is provided below:

```
<?xml version="1.0" encoding="utf-8"?><GetFeature
xmlns="http://www.opengis.net/wfs"
xmlns:cp="http://thecarbonproject.com"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:gml="http://www.opengis.net/gml" service="WFS" version="1.1.0"
outputFormat="text/xml; subtype=gml/3.1.1" maxFeatures="50000"
handle="" ><Query typeName="cp:Haiti_Buildings" srsName="EPSG:4326"
><ogc:Filter><ogc:Within><ogc:PropertyName>shape</ogc:PropertyName><gml
:Envelope srsName="EPSG:4326"
xmlns:gml="http://www.opengis.net/gml"><gml:lowerCorner>-
72.3692407366058 18.5340953146098</gml:lowerCorner><gml:upperCorner>-
72.2358722981114
18.5966590713815</gml:upperCorner></gml:Envelope></ogc:Within></ogc:Fil
ter></Query></GetFeature>
```

Test queries were run with BBOX parameters that covered large and larger geographic regions and increasing MaxFeatures, resulting in increasing volumes of GML streaming to the Gaia application as shown in the figure below.

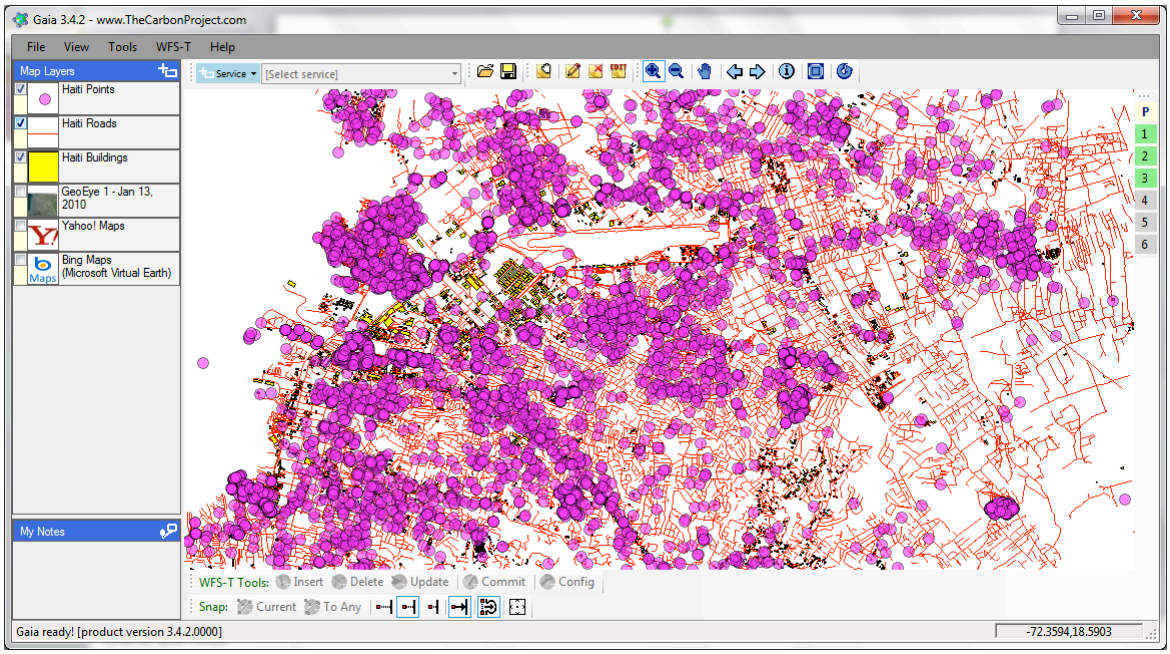

Figure 3 - GML streaming to the Gaia application during testing

Since applications such as Gaia include functions for both accessing and rendering GML data simultaneously (as shown in Figures 2 and 3) performance metrics were obtained for data access only by using test applications from The Carbon Project that request and retrieve the data, but do not render it.

The following GML FeatureTypes were included in the testing, each FeatureType represented the total amount of OSM data (as GML) available for over Haiti and the Dominican Republic -

- Haiti Points
- Haiti Roads
- Haiti Buildings

The maxFeatures property on the request sequence was set to 100,000 features for each FeatureType. The timestamp indicated in Table 2 below is the time it took to make the request and parse it into usable objects without rendering. All times listed are the averages calculated by making multiple requests for streaming GML.

Table 2 - Metrics on accessing test data using GML streaming

FeatureType Name	GML Geometry	GML Feature Count	Time to Access and Download from Cloud
Haiti Points	Point	16,717	2.15 seconds
Haiti Roads	MultiLineString	93,306	32 seconds
Haiti Buildings	MultiPolygon	100,000	24 seconds

Both the Haiti Buildings and Haiti Points FeatureType had four properties each (including Geometry). The Haiti Roads FeatureType had had seven properties (including Geometry). The GML data was deployed on WFS on the Microsoft Azure Cloud according to the GML simple features (GML-SF) profile. The Internet connection used was a commercial commodity high-speed connection. The computer supporting the test

client was a two year old commodity desktop system running Microsoft Windows software.

Additional performance metrics were obtained for data access by GML streaming using test applications from The Carbon Project that request and retrieve the data, and insert the data into a SpatiaLite database as described in Section 8 below.

7.3 Testing Access with OGC Mobile Client Applications

Data access with GML Streaming was also tested on a Mobile app for Android operating system. In this test the WFS with Streaming GML was registered on the CarbonCloud Sync application service. Mobile users were then able to communicate with a CarbonCloud Sync application service via OGC WFS operations and access features across an open standard WFS interface suitable for mobile users.

For example, the figure below shows the app accessing GML test data for the Haiti Buildings feature type.

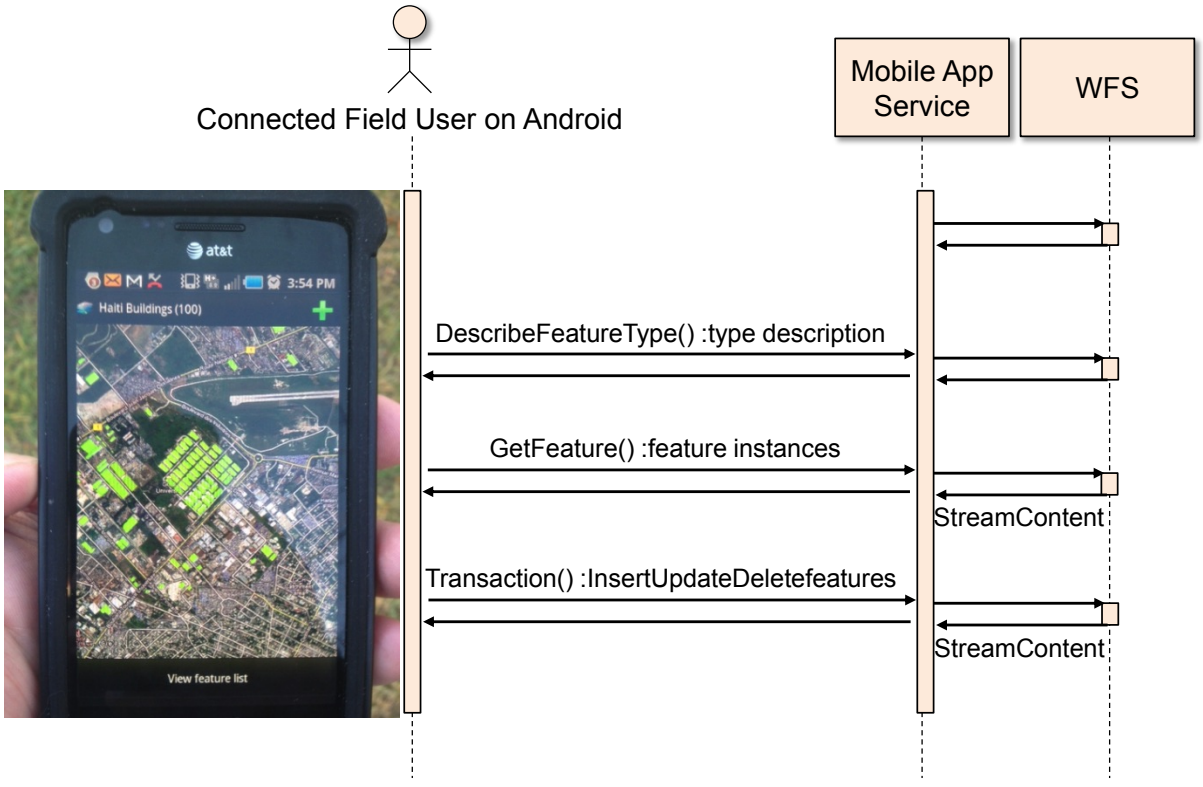

Figure 4 - GML Streaming to a Field User on The Carbon Project Android app

7.4 Transaction Handling in the Basic Workflow for Streaming GML

While the basic workflow supports reading features from the service, WFS may also support the ability to perform data update operations via Transactions operations. The Web Feature Service Transactional (WFS-T) standard defines a set of operations that include insert, delete, and update of features. In most cases the changes will be published and be available to other end-users of the services. Transaction operations were tested with obtained Point, MultiLineString and MultiPolygon GML data and the mobile application.

7.5 Other Client Applications

Streaming GML access was tested on a Silverlight web application developed by Lockheed Martin utilizing the ESRI ArcGIS API and custom built client libraries (Figure 5). This application allows the user to navigate the map, search for GIS data (i.e. ArcGIS services, WFS, WMS, WMTS, KML, and GeoRSS) and then combine data from different sources on the map to create a common operating picture.

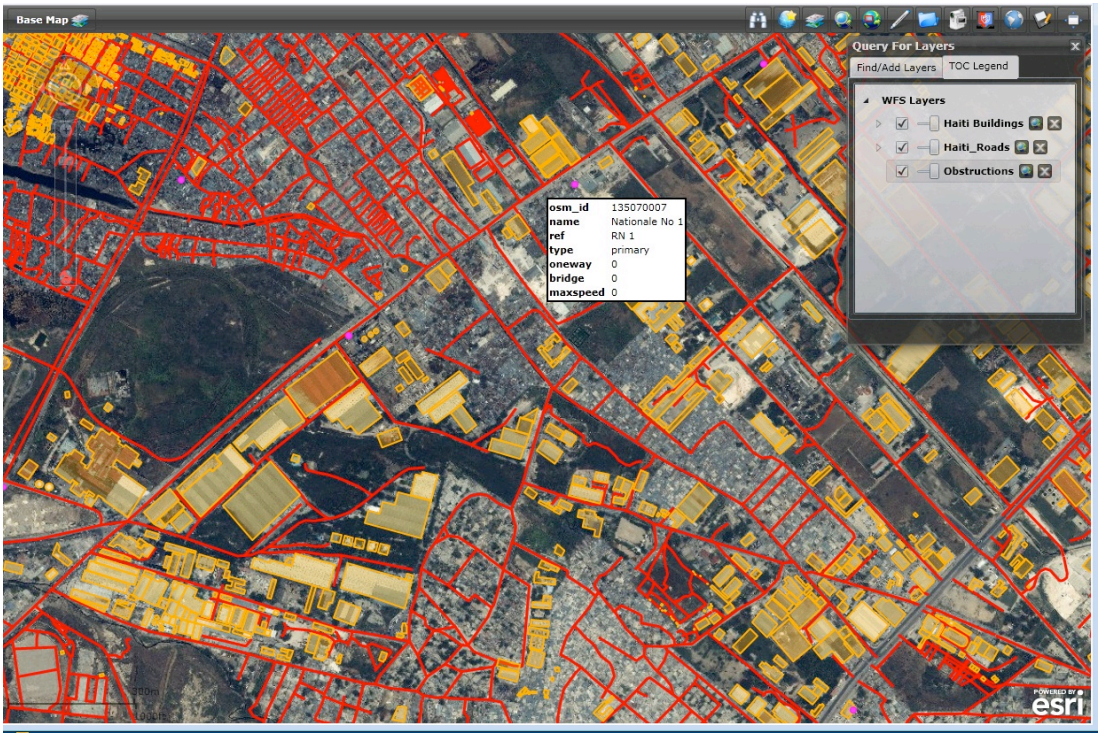

Figure 5 - GML Streaming accessed by Lockheed Martin application

In addition, Streaming GML access was successfully tested on the ESRI ArcGIS 10.0 desktop. Access required the setting of a number of parameters in the desktop application, however, once set GML data access was seamless.

8 SQLite/SpatiaLite Workflow for GML Streaming

Mobile device users who require map/geospatial application services and operate in disconnected or limited network connectivity environments are often challenged by limited storage capacity and the lack of open format geospatial data to support these applications. The current situation is that each map/geospatial application requires its own potentially proprietary geospatial data store. These separate application-specific data stores may contain the same geospatial data, wasting the limited storage available, and requiring custom applications for data translation, replication, and synchronization to enable different map/geospatial applications to share the same world view. In addition, many existing geospatial data stores are platform-specific, which means that users with different platforms must translate data to share it.

This section describes how the OWS-9 effort assessed, developed and tested the delivery of large payloads consisting of geospatial data sets and/or collections of data sets to SpatiaLite to store the data for use by mobile applications using GML.

8.1 SQLite/SpatiaLite and GeoPackage Overview

SpatiaLite is an open source library that extends the SQLite core to support Spatial SQL capabilities. SQLite is a software library that implements a self-contained, serverless, transactional SQL database engine. SpatiaLite implementations described in this document use the draft GeoPackage specification.

The draft OpenGIS® GeoPackage Implementation Specification defines a GeoPackage as 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, especially on mobile / hand held devices in disconnected or limited network connected environments.

Direct use requirements include the ability to access and update data in a "native" format without intermediate format translations in an environment (e.g. through an API) that guarantees data model and data set integrity and identical access and update results in response to identical requests from different client applications. Specifically, a GeoPackage can contain multiple vector feature types, rasters from various sources, and multiple tile matrix pyramids. A GeoPackage supports storage of rasters and tiles in multiple specified image file formats. Tiles are expected to be georectified or orthorectified view-space images, while rasters could also be raw "as collected" images. An individual GPKG may contain one, some or all of these types of geospatial data. The GeoPackage API provides Simple Features SQL access to vector features and geometries, and additional SQL functions on Rasters, Tiles, and their descriptive metadata. The GeoPackage API supports implementation of data content management and integrity constraints via SQL triggers.

The GeoPackage Manifest serves as a table of contents and data source access metadata store for the contents of the GeoPackage data container. The Carbon Project's effort's in OWS-9 extended this aspect of GeoPackage since easy methods were needed by mobile app developers to help determine what data was in each SQLite column as described in the following section. .

8.2 Creating GeoPackage from WFS

The streaming approach may be used to create GeoPackages from WFS serving a set of features encoded as GML. As part of this project a 'GeoPackage Manager' was developed to access the cloud-based WFS established by The Carbon Project and support testing this workflow.

Figure 6 and the following descriptions illustrate the sequence of operation calls to initialize a GeoPackage, create feature tables in SQLite, obtain GML as streamed data from a datastore across the WFS interface and populate the GeoPackage.

1. A GeoPackage Manager component initializes a GeoPackage on SQLite.
2. A GeoPackage Manager requests a capabilities document from the WFS.
3. A GeoPackage Manager component creates feature tables in GeoPackage.
4. A GeoPackage Manager component imports GML into GeoPackage.

Based on this workflow the OWS-9 GeoPackage Manager component implemented by The Carbon Project provided the ability to initialize a GeoPackage on SQLite from a Streaming GML source implementing the OGC Web Feature Service interface, and load Streaming GML data into it.

Future versions of the GeoPackage specification and/or one for a GeoPackage Web Service may address additional utilities for creating GeoPackages and importing vector, raster and tile data in various formats.

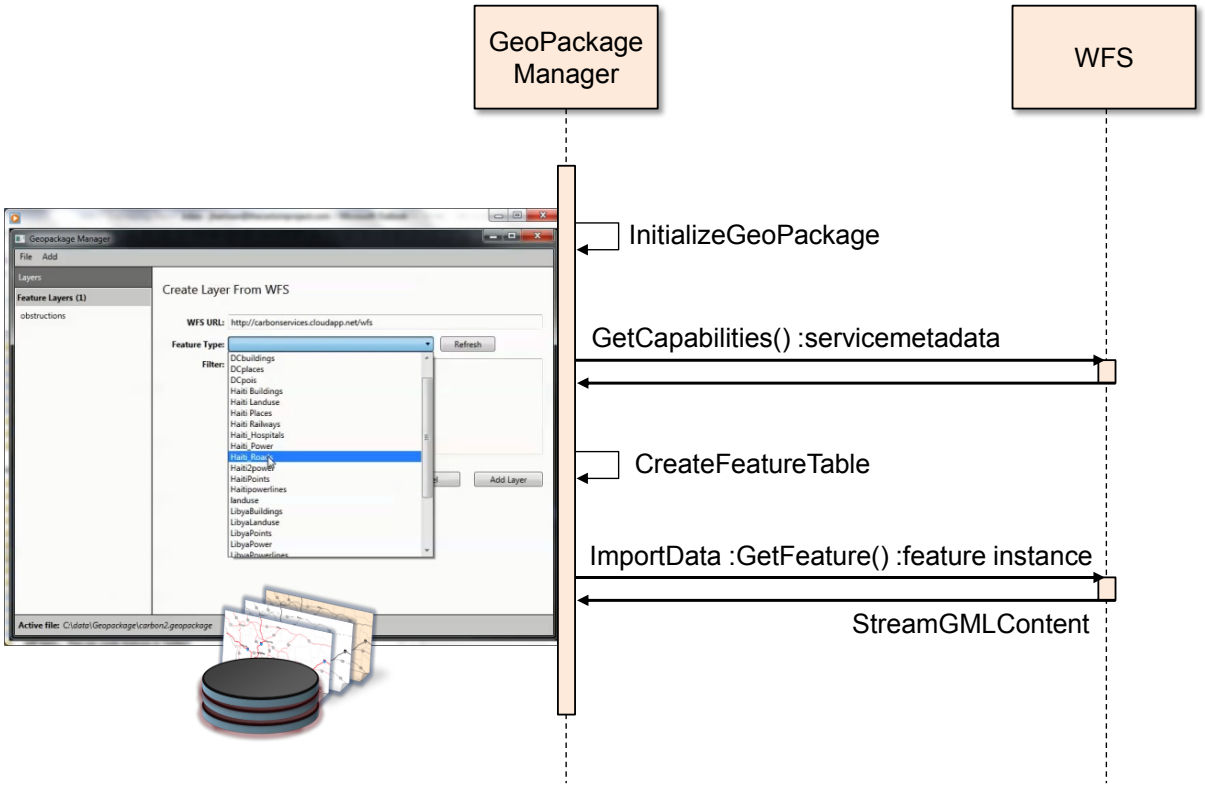

Figure 6 - Creating a GeoPackage from Streaming GML using The Carbon Project GeoPackage Manager component

During the implementation process it was assessed that stores data in an ambiguous way. Specifically, the question was raised – "How does an application, including a mobile application, figure out what data is in each column?" For instance, SQLite has a numeric affinity, and Boolean, Datetimes and decimal values should use a numeric SQLite affinity. What we not clear when integration experiments commenced was - how can an application determine what each column is from a client program that has nothing to go by other than the SQLite file. The client may read the column as a numeric, but it could really be either a Boolean, Datetime or decimal value stored. In other words, methods were needed to figure out the exact datatype of each column. To address this during testing The Carbon Project developed a simple XML description which led to extensions of the GeoPackage draft and development of more standardized mechanisms. A sample of this XML description is included below.

```
<?xml version="1.0" encoding="UTF-8"?>
<cp:CarbonMetadata xmlns:cp="
[[http://www.thecarbonproject.com/][http://www.thecarbonproject.com]]">
<cp:WfsUrl>http://somecloud.somecloudapplication.net/wfs</cp:WfsUrl>
<cp:FeatureType>cp:HaitiPoints</cp:FeatureType>
<cp:Filter/>
<cp:Fields xmlns:cp="
[[http://www.thecarbonproject.com/][http://www.thecarbonproject.com]]">
  <cp:Field xmlns:cp="
[[http://www.thecarbonproject.com/][http://www.thecarbonproject.com]]">
   <cp:Name>shape</cp:Name>
   <cp:Title>shape</cp:Title>
   <cp:Description/>
   <cp:FieldType>Point</cp:FieldType>
   <cp:IsRequired>False</cp:IsRequired>
  </cp:Field>
  <cp:Field xmlns:cp="
[[http://www.thecarbonproject.com/][http://www.thecarbonproject.com]]">
   <cp:Name>osm_id</cp:Name>
   <cp:Title>osm_id</cp:Title>
   <cp:Description/>
   <cp:FieldType>Integer</cp:FieldType>
   <cp:IsRequired>False</cp:IsRequired>
  </cp:Field>
  <cp:Field xmlns:cp="
[[http://www.thecarbonproject.com/][http://www.thecarbonproject.com]]">
   <cp:Name>timestamp</cp:Name>
   <cp:Title>timestamp</cp:Title>
   <cp:Description/>
   <cp:FieldType>String</cp:FieldType>
   <cp:MaxLength>20</cp:MaxLength>
   <cp:IsRequired>False</cp:IsRequired>
  </cp:Field>
  <cp:Field xmlns:cp="
[[http://www.thecarbonproject.com/][http://www.thecarbonproject.com]]">
   <cp:Name>name</cp:Name>
   <cp:Title>name</cp:Title>
   <cp:Description/>
   <cp:FieldType>String</cp:FieldType>
   <cp:MaxLength>48</cp:MaxLength>
   <cp:IsRequired>False</cp:IsRequired>
  </cp:Field>
  <cp:Field xmlns:cp="
[[http://www.thecarbonproject.com/][http://www.thecarbonproject.com]]">
   <cp:Name>type</cp:Name>
   <cp:Title>type</cp:Title>
   <cp:Description/>
   <cp:FieldType>String</cp:FieldType>
   <cp:MaxLength>16</cp:MaxLength>
   <cp:IsRequired>False</cp:IsRequired>
  </cp:Field>
```

```
</cp:Fields>
</cp:CarbonMetadata>
```

The GeoPackage Manager component was also leveraged to obtain additional performance metrics for data access by GML streaming to request and retrieve the data, and insert the data into a SpatiaLite database, as shown in the last column of the table below. Both the Haiti Buildings and Haiti Points FeatureType had four properties each (including Geometry). The Haiti Roads FeatureType had had seven properties (including Geometry).

Performance metrics on the time to insert the obtained GML data into a SpatiaLite database are shown in the table below:

Table 3 - Time to insert GML into SQLite/GeoPackage for use on Mobile devices

FeatureType Name	GML Geometry	GML Feature Count	Time to Access and Obtain GML	Time to Insert GML into GeoPackage
Haiti Points	Point	16,717	2.15 seconds	5 seconds
Haiti Roads	MultiLineString	93,306	32 seconds	38 seconds
Haiti Buildings	MultiPolygon	100,000	24 seconds	32 seconds

8.3 Using GeoPackage for Connected/Disconnected Mobile Apps

A GeoPackage mobile app was created to access and update the Streaming GML imported into a GeoPackage, as shown in Figure 7 below.

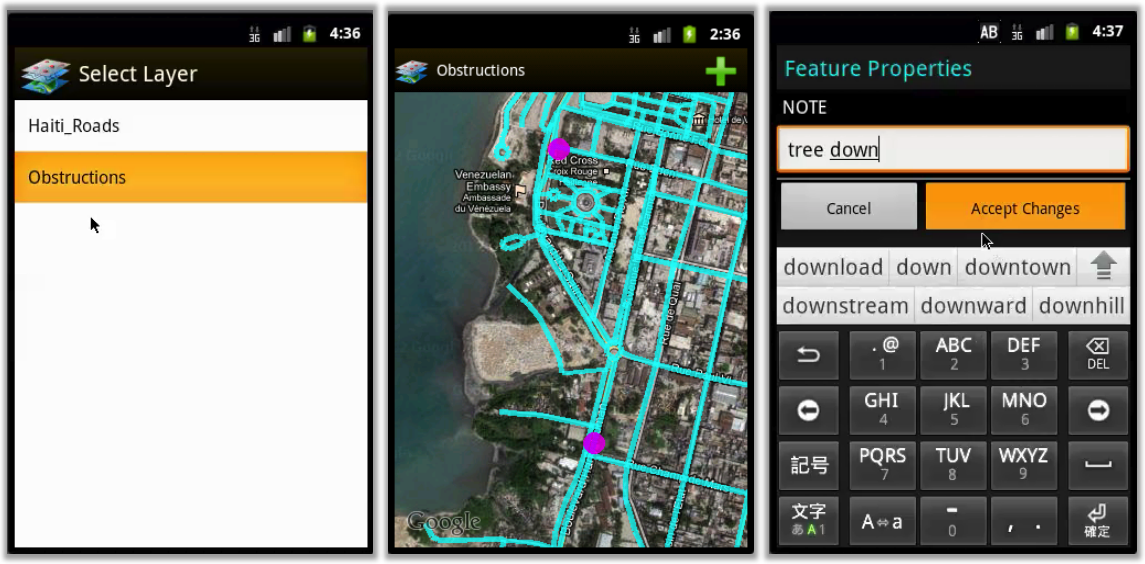

Figure 7 - GeoPackage mobile app on Android

In this Integration Experiment, data in the GeoPackage created from Streaming GML feature data sources described in Section 7 was transferred to an Android mobile application from The Carbon Project. Once the GeoPackage was loaded this mobile application provided the capability to access feature data from the GeoPackage, Pan, Zoom, Select Layers, access data, and edit (Inert, Update, Delete) Feature Geometries and Properties.

For this project The Carbon Project's mobile application made Transactions against the SQLite/SpatiaLite and the Streaming GML services (exposed as WFS). In this approach, the features and changes are applied to the main table in GeoPackage. Transactions are captured in a separate table, so the application knows what has changed. As a sync occurs, the transaction table gets emptied and may be sent to a GeoSynchronization service or the WFS. In this approach, transactions are stored as the feature type id, feature id, operation and XML in a new table. The XML is WFS Transactions XML. When a sync is performed, a WFS:Transaction is created containing all the Inserts/Updates/Deletes in the table, and then sent. The table is emptied if the Transaction was successful. The application code handles instances in which multiple changes occur to the same feature before a sync occurs. For instance, if the user Inserts a new feature and then edits it, all that is sent is a single Insert to the WFS.

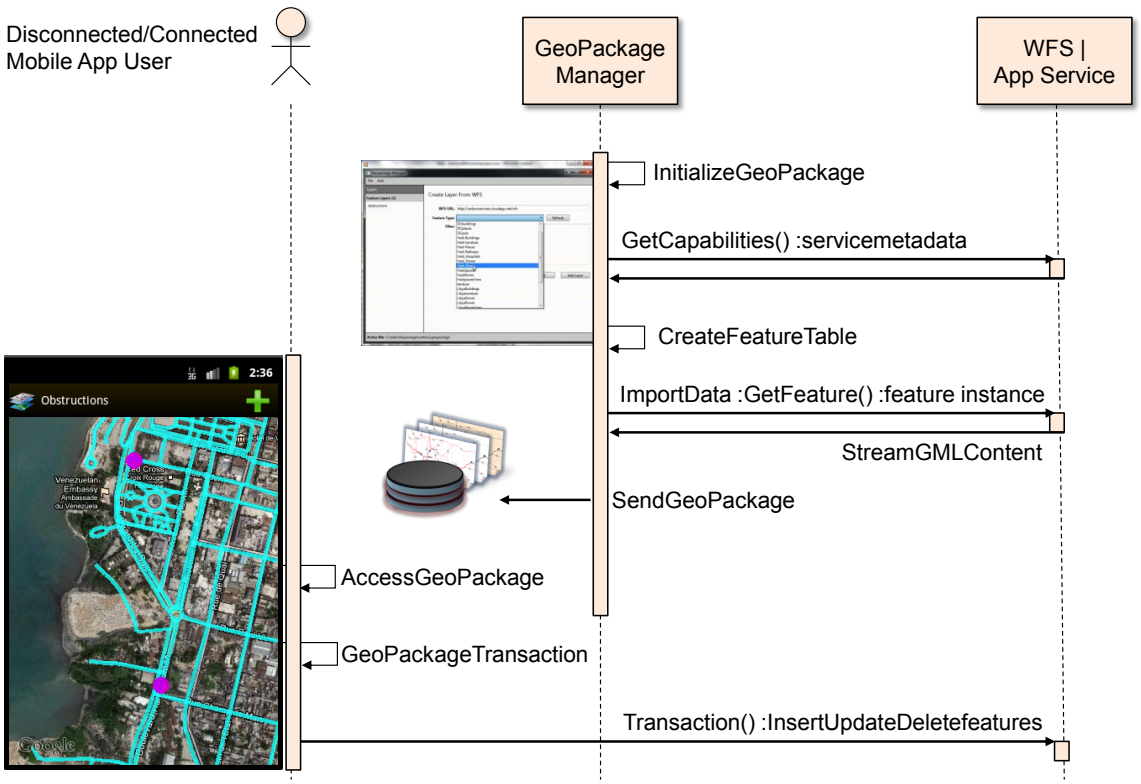

Figure 8 – Sequence Diagram for GeoPackage creation from Streaming GML, and use on mobile application

9 Recommendations

Key recommendations for future work include

- Testing the Streaming GML approach should be tested with OGC Web Feature Services (WFS) delivering Geography Markup Language (GML; ISO 19136) enhanced with "Resolution Parameters" on the GetFeature Request to support data access client applications using density-specific content models.

- Transactions into GeoPackage.
- GeoSynchronization with Streaming GML and GeoPackages.

Of particular utility to the dissemination of large payloads consisting of geospatial data sets and/or collections of data sets between machines connected via a network is OGC Change Request (CR) 132, OGC Document 11-004. This CR recommends 'Adding resolution parameters to GetFeatureRequest to support visualization clients' using Web Feature Service (WFS) Implementation Specification 2.0. The scope of this CR covers the addition of three optional parameters to the GetFeature request. The parameters are called: resolution, resolutionX and resolutionY. In this approach, you either specify resolution OR resolutionX/resolutionY. If you specify resolution this implies this means that the resolution in the X and Y directions is that same. Otherwise you can set the X and Y resolution independently. The resolution parameters may be used to generalize geometry suitable for the specified resolution. This means that non-visible segments are removed or it may even mean not displaying anything if you are zoomed too far out.

Based on the lessons learned in OWS-9 the "Resolution Parameters" on the GetFeature Request described may be extended to support data access client applications using density-specific content models where specific feature types or resolutions of feature types are provided at differing resolution levels by Streaming GML services.