Feasibility of OGC’s GeoSynchronization Service for delivering the incremental updates from a national topographic database

Eero Hietanen, Lassi Lehto and Pekka Latvala
Finnish Geospatial Research Institute (FGI)
National Land Survey of Finland
Geodeetinrinne 2
02430 Masala, Finland
eero.hietanen@nls.fi

Abstract

Increasing use of the geospatial data has led to the situation, where automatic processes for updating the data are needed. One method to implement a process of incremental update is OGC’s GeoSynchronization Service (GSS). We have piloted GSS in a multi-provider setting, where Finnish Environment Institute provides spatial data set for a hydrologic network, which is based on the geometries produced by the National Land Survey of Finland (NLS). In this paper, we investigate the feasibility of a Web Service based on GSS to deliver the incremental updates from NLS’s topographic database to service providers in a standardised format.
Keywords: incremental update, GeoSynchronization, Web Feature Service Transaction, service database.

1 Introduction

The increasing trend of opening governmental data sets in Europe has positively affected the provision of geospatial services. Often the service provider (i.e. the user of the governmental data sets) fetches data sets from the download services of the data providers and adds some additional data on the top of those data sets. Sometimes, a lot of effort is put on the additional data, which uses data sets as a spatial framework. The value of the additional data and the provided service is dependent on the accuracy and quality of the used framework data sets. Thus, the update process of the used data sets is vital for the quality of the provided services.

Data producers provide these updates typically in bulk, which means that the old data set is replaced with a new one as a whole. This procedure causes enormous update processes for the service providers, who have to relink their own data to the objects in the new framework data set. It also leads to lower update frequency, hence worsening service quality.

Services built on top of national data sets in Europe include Web Services obliged by the INSPIRE directive [3] and defined by the INSPIRE regulations for implementing the Network Service [2]. The data themes introduced in the INSPIRE directive often involve data from multiple data providers, thus a separate service database to provide the INSPIRE compliant service is needed [4].

All of this has led to the situation, where automatic processes for updating the data are needed. In incremental updating, only the changed data from the original data set is delivered to the service providers, in a way that it can be feasibly used to update the external data set [1].

One method of implementing incremental update is Open Geospatial Consortium’s (OGC) GeoSynchronization Service (GSS) candidate standard [8]. GSS provides guidelines to serve changes data in standardized format over the network. We have piloted GSS in the context of the INSPIRE directive. The case in the pilot was that Finnish Environment Institute (SYKE) provides spatial data set for a hydrologic network, which is based on the geometries produced by the National Land Survey of Finland (NLS). [5]

To use GSS specification for serving the incremental updates, we had to design a database model to manage the changes data. In addition, we need an interpreter for deriving the changed features in NLS’s topographic database to the actual changes data. In this paper, we investigate the possibilities of a Web Service based on GSS to deliver the incremental updates from NLS’s topographic database to the service providers in standardised format.

Section 2 of this paper presents the principles of OGC’s GeoSynchronization Service. Section 3 presents the principles of the designed GSS Service Module. Section 3.1 describes Changes Data Interpreter, which derives the actual changes data from the changed features in NLS topographic database. Section 3.2 describes the database model used to manage the changes data. Section 3.3 presents the implementation of the actual GSS component, which allows the changes to be queried. Section 4 discusses the possibilities and advantages of GSS Service Module to provide the changes data of NLS’s topographic database.

2 OGC’s GeoSynchronization Service

The basic idea of the GSS candidate standard is that all the data collectors can propose changes to the features of the data provider. The proposed changes are reviewed in the service and either accepted or rejected. GSS provides the changes data for the data users as events. The user can subscribe to the service for automatic updates or use the query API, which allow all the events to be queried. [8]

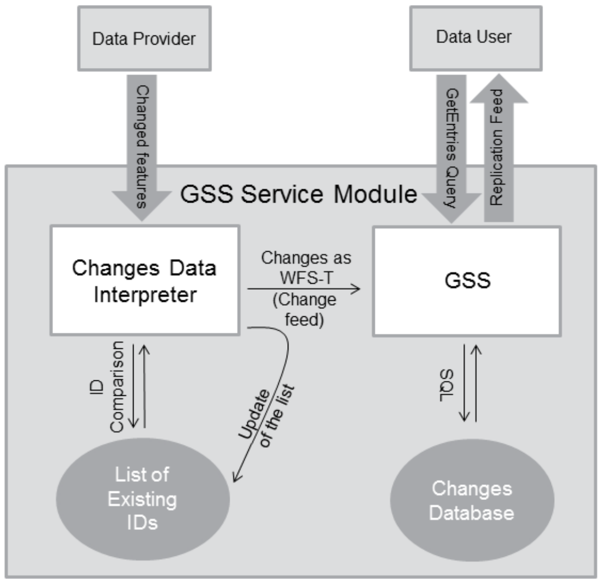
There are three different feeds defined to implement GSS in the candidate standard. A Proposed Change feed (later Change Feed), Resolution feed and Replication feed. The feeds are defined in ATOM feed format [6]. Change feed is for the proposed changes from the data collectors to GSS and Resolution feed for the response, indicating if the proposed changes are accepted. Replication feed is for serving the changes data to the users. In our study, we are dealing with authorized changes made by the data provider, thus Resolution feed is not investigated in this paper.

The candidate standard of GSS describes a query method for Replication feed. The defined query operator is GetEntries, which allows the events to be queried using spatial or non-spatial predicates. The response to this query operation is custom-made Replication feed containing all the events satisfying the query predicates.

3 GSS Service Module

Implementation of the GSS Service Module consists of four components: Changes Data Interpreter, Table of existing feature IDs, Changes Database and GSS component (Figure 1).

Figure 1: Data flow of GSS Service Module



In the first phase of the process, a list of the IDs of the features existing at a certain moment in the data provider's data set is stored to the database. GSS Service Module stores and provides all the change information from the time after that moment. The changed features are fed to the Changes Data Interpreter, which creates the Change feed (Section 3.1). The Change feed is sent to the GSS component (Section 3.3), which controls the Changes Database (Section 3.2) and serves the Replication feed to the data users.

3.1 Changes Data Interpreter

The data of NLS's Topographic database are stored in the Smallworld environment. The database has all the history features, but there are no links between the new and old features. The life spans of the feature objects are managed with three separate attribute information: unique ID, begin of life span and end of life span. If a new feature is added to the database, the feature will have new values for the unique ID and for the begin of life span attributes. If the feature is deleted, it will keep the same ID and have a value for the end of life span attribute. If the feature is updated, it will keep the same ID and have new value for the begin of life span attribute. In addition, a history feature with new values for the ID and for the end of life span attributes is made to represent the feature before the change. All the changed features, including new, updated and deleted ones, can be fetched from the database using time constraints.

Changed features can be retrieved from Smallworld environment in a text based format. However, any other generic format, which can be handled with GIS tools, would be fine. To interpret the type of the change of the particular feature, we need a table of IDs of the features existing in the NLS's topographic database before the changes (Table of existing feature IDs). There are three types of change operations we are interested in: INSERT, UPDATE and DELETE. Those are operations of Web Feature Service (WFS) Transaction [7]. When the IDs of the changed features are compared with the table of existing IDs, corresponding WFS-T operations can be derived:

- Feature has an old ID and has not the end of life span attribute: Transaction operation is UPDATE.
- Feature has an old ID and has the end of life span attribute: Transaction operation is DELETE.
- Feature has a new ID and has not the end of life span attribute: Transaction operation is INSERT.
- Feature has a new ID and has the end of life span attribute: Recognized as a history feature.

The Changes Data Interpreter is implemented with Python programming language using Django Web framework. This component processes the changed features data, produces the correspondent WFS Transaction operation and sends the data to GSS.

3.2 Changes Database

There are multiple ways to store the versioning data. In the prototype, the Changes database was designed to be simple. It constitutes of one single database table, where one record corresponds one change event of a single feature. Table 1 describes the columns of the database table.

Table 1: Structure of the Changes Database Table.

Column	Data Type	Description
GID	integer	Unique identifier of the feature
Type	string	Name of the feature class
Change Time	date/time	Timestamp of the change
Operation	string	INSERT/UPDATE/DELETE
Transaction	string	The change, encoded as WFS Transaction operation
Geometry	geometry	Geometry of the feature after the change. If the operation is DELETE, then before the change.
EntryID	integer	Unique identifier of the change event
Author	string	Publisher of the change

This simple solution enables efficient queries based on time of change, feature type, feature ID and the area of interest. Transaction column is implemented to reduce the encoding time, when formatting Replication feed according to the query. Although, all the information needed for WFS Transaction operation is already in the table, Transaction column is implemented to reduce the encoding time when formatting Replication feed according to the query.

3.3 GSS implementation

In the prototype, GSS component is implemented as a Java Servlet. Two main functionalities are to process the Change feed and to provide the Replication feed. GSS receives the incoming Change feed as WFS-T encoded HTTP POST request, parses the contained information and stores it to the Changes Database. Change feed is not implemented according to the OGC's GSS candidate standard, because in the studied update process it is used only as an inner procedure of the GSS Service Module between the Changes Data Interpreter and the GSS component. Since WFS-T DELETE operation does not include the geometry of the feature, we need another solution to add the geometry of the deleted feature into the Changes Database. One option is to add separate geometry information to the Change feed, in addition to WFS-T operation.

In another role, the implemented GSS component works as a query API, which returns Replication feed in ATOM format. The query operation according to GSS candidate standard is GetEntries and it works as a HTTP GET request. Events can be filtered with the ID of an event or a feature and with temporal and spatial filters. Although FEATUREID filtering parameter is not according to the GSS candidate standard, it was added to feasibly get the whole change history of a certain feature.

The GetEntries query returns a custom-made Replication feed, including the filtered events. The response is encoded in ATOM feed format according to the GSS candidate standard. Users of the service can use the WFS Transaction operation, encoded inside the ATOM content to update their own data sets. In addition to GetEntries operation, there is a custom-made GetEntryIdObjectGeometry operation to get the geometry of a certain change event. This feature was seen useful especially when exploring the change events provided by a GSS in a map application.

4 Advantages of the GSS Service Module

The prototype implementation of GSS Service Module has been developed for the specific update process between two data providers. In the demonstration, changed features, which were fed to the Changes Data Interpreted, were created with an OpenLayers based client application. Another OpenLayers based client application was used to demonstrate the updating of a hydrologic network according to the Replication feed. The designed process is found to be appropriate for this particular case. However, this kind of GSS Service Module, which serves incremental updates, has much more application possibilities. Introduced solution works both for a whole dataset, when larger amount of features need to be synchronized at once, and for a more limited dataset, where manual controlling is needed and features are updated one-by-one.

The described Changes Database implementation with GSS query API provides access to the whole version history of the source data set. Version history would be complete, if the original version of the data set was also available through the service. It would allow deriving all the different states of the data set during its history. Full version history gives service providers an opportunity to choose the most appropriate update frequency for their service databases.

The introduced GSS Service Module solution does not solve the problem of appearing or disappearing IDs, which emerges when separate features are merged to one or one feature is divided into two. This will be part of the further studies on the subject. Another need for further investigations is enabling the synchronization of different schema structures, scaling the service to large amounts of data and finding methods for ensuring the correctness of the update processes.

5 Acknowledgment

The work described in this paper has been co-funded by the National Land Survey of Finland, the Finnish Environmental Institute and the Finnish Ministry of Agriculture and Forestry.

References

[1] A. Cooper and A. Peled. Incremental Updating and Versioning. *20th International Cartographic Conference, Beijing, Vol 4, pp. 2804-2809*, 2001

[2] European Commission. COMMISSION REGULATION (EC) No 976/2009 of 19 October 2009 implementing Directive 2007/2/EC of the European Parliament and of the Council as regards the Network Services, 2009. Available at: <http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:02009R0976-20101228&from=EN> [Accessed: 2015-01-16].

[3] European Commission. INSPIRE Directive, 2007. Available at: <http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2007:108:0001:0014:EN:PDF> [Accessed: 2015-01-16].

[4] E. Hietanen and L. Lehto. OS Implementation of INSPIRE Hydrography Services in a Multi-Provider Setting. In INSPIRE Conference, 18-20 June, 2014, Aalborg, Denmark. Available at: <http://inspire.ec.europa.eu/events/conferences/inspire_2014/schedule/submissions/261.html> [Accessed: 2015-01-16]

[5] L. Lehto, E. Hietanen and P. Latvala. Using Geosynchronization for Incremental Update of INSPIRE Service Databases. In The Sixth International Conference on Advanced Geographic Information Systems, Applications and Services, GEOProcessing 2015, Feb 22 – 27, 2015, Lisbon, Portugal. (Unpublished).

[6] M. Nottingham and R. Sayre, editors. The Atom Syndication Format. The Internet Society, 2005. Available at: <http://tools.ietf.org/html/rfc4287> [Accessed: 2015-01-16].

[7] P. A. Vretanos, editor. OpenGIS Web Feature Service 2.0 Interface Standard. Open Geospatial Consortium, 2011. Available at: <http://portal.opengeospatial.org/files/?artifact_id=39967> [Accessed: 2015-01-16]

[8] P. A. Vretanos, editor. OWS 7 Engineering Report – Geosynchronization Service. Open Geospatial Consortium, 2011. Available at: <http://portal.opengeospatial.org/files/?artifact_id=39476> [Accessed: 2015-01-16].