Incremental Update in Web Service Environment – Case: Use of the OGC's GeoSynchronization Service to Integrate Hydrographic Datasets

Pekka Latvala, Eero Hietanen

Finnish Geospatial Research Institute (FGI), National Land Survey of Finland, pekka.latvala@nls.fi, eero.hietanen@nls.fi

Abstract. The increasing use of local copies of spatial data sets that are maintained by other parties as source data for various web services sets demands for data update procedures. The copied data sets should be kept continuously up-to-date in order to maintain good data quality in the services. This paper describes a piloted example case where a prototype implementation of the Open Geospatial Consortium's (OGC) GeoSynchronization Service (GSS) candidate standard was created in order to execute the same updates that were carried out in a master hydrographic data set in a local copy of that database.

Keywords: GeoSynchronization service, web service

1. Introduction

The current trend in Europe where different governmental organizations are opening up their spatial data sets has led to increased use of spatial data and services. This trend has been largely set in motion by the INSPIRE directive (European Commission, 2007) that sets requirements for the INSPIRE participants to set up various types of Web Services that cover data from multiple themes. Different companies, organizations and individuals can download the original data sets from the data providers and create their own products and services on top of them. The copied data sets should be updated regularly from the master databases in order to maintain good data quality in the services. The execution of a full update where all data are replaced is often unfeasible because of the large amounts of data or because the data might have been changed or new data might have been added in the local service databases. An alternative to full update is to perform the updates incrementally, feature-by-feature. The GeoSynchronization Service (GSS) specification developed by the Open Geospatial Consortium (OGC) defines a web service interface that can be used for executing the data updates at feature level.

Previous research on the use of GeoSynchronization for integrating data sets has been made by (Milanović et al. 2010) who has studied automated synchronizing of geographic data in Catalonia, (Sparks et al. 2011) who has studied integration of hydrographic datasets in Indiana and (Chormann & Harrison 2012) who have examined the use of GeoSynchronization in New Hampshire.

This paper describes a piloted example case for GSS implementation where a prototype of the GSS service was created for handling the updates for hydrographic data. The second chapter introduces the GeoSynchronization service specification. The third chapter describes the studied example case. The fourth chapter describes the prototype GSS implementation. The paper ends with discussion and conclusions.

2. GeoSynchronization Service

The OGC's GSS specification provides one solution to the incremental data update problem. The GSS specification is currently an OGC candidate standard and it has not yet achieved a full standard status. The latest activities related to the standardization process of the GSS have been executed in 2011 (Vretanos 2011a, Vretanos 2011b).

The workflow of the GSS is based on the use of three different Atom feeds (Nottingham & Sayre 2005) (*Figure 1*) that deliver information on the requested change proposals, the accepted changes and the changes that have been executed on the master database. The GSS defines three user roles: publisher, reviewer and subscriber. Publisher can suggest change proposals that are to be executed in the master database. These change proposals are collected into the change feed (1) that is sent to reviewer (2) who inspects the change proposals and either accepts or rejects them. The reviewer's decisions are collected into the resolution feed (3) that is sent back to the publisher (4) in order to deliver information about the acceptance of the change proposals. The accepted changes are executed in the database (5). The change proposal reviewing process can be executed either manually or algorithmically depending on its content and requirements. Alternatively, it is also possible to skip the change proposal reviewing phase entirely and execute the proposed changes immediately in the master database if the publisher is viewed as trusted authority.

The changes that are executed in the database are collected into the replication feed (6) that allows the subscribers to synchronize their local databases with the master database. The subscribers can read the replication feed (7) and use it for executing the exact same changes in their local database (8). The subscribers can also query the replication feed and fetch only those changes that are relevant to them.

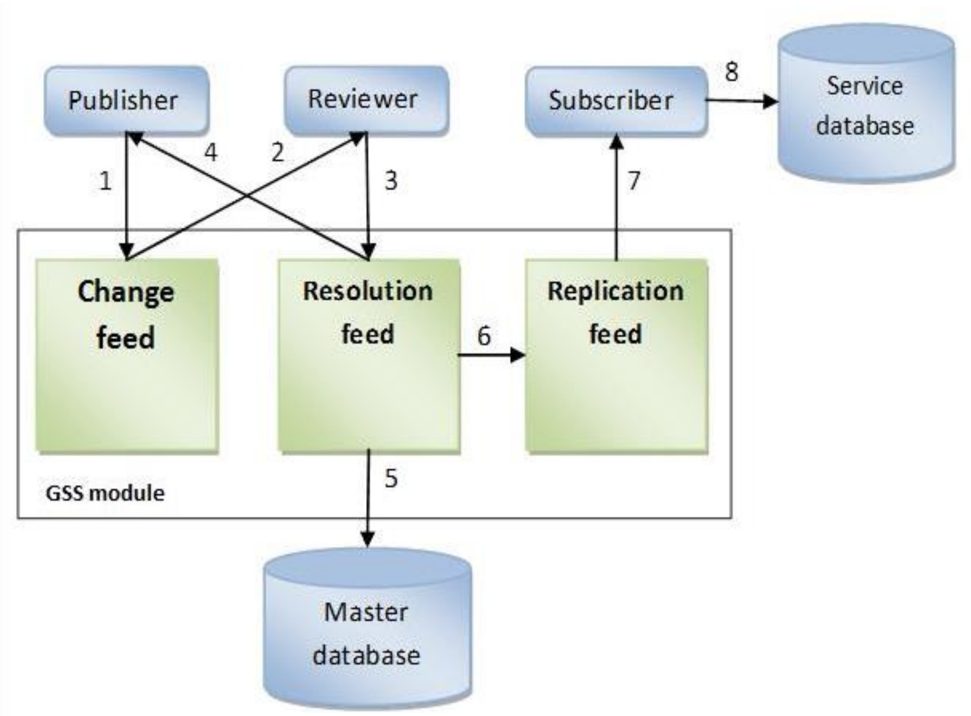

Figure 1: GSS workflow

3. Studied Example Case

The use of GeoSynchronization Service was studied in an example case where the updates that are executed in a master database are wanted to be replicated in a copy of that database. The example case is related to the handling of updates for hydrographic data in a situation where the data is collected and maintained by the National Land Survey of Finland (NLS) and used by the Finnish Environmental Institute (SYKE). SYKE has copied the hydrographic data from the NLS between the years 2000 and 2008 and created a hydrologic network model from the hydrographic features. After the initial data load, SYKE has performed reclassification and topology checks for the data and updated it partially. Most of the hydrographic data has not been updated since the initial data load and it has fallen out of sync with the master database. SYKE has identified a strong need for keeping their network model up-to-date by replicating regularly the changes that are made to the original database into their local database.

4. GSS Prototype Implementation

The GSS prototype implementation was created as a Java Servlet that was deployed to the Apache Tomcat application server. The GSS candidate standard specification was implemented partially. The implemented operations, listed in *Table 1,* were: *Insert*, *Update, Delete, GetEntries* and the custom *GetEntryObjectGeometry operation*. The change proposal reviewing stage was bypassed in the prototype because all proposed changes are seen as authoritative changes that don't need to be reviewed and they are accepted immediately. Therefore, the operations: *Accept Change, Reject Change* and *Review Changes* that are related to the change proposal reviewing were not implemented in the prototype. Also the operations that are related to topics and subscriptions were not implemented in the GSS prototype because the resolution feed querying functionality via the *GetEntries* operation was considered to be sufficient for the prototype.

GSS operation	**Implemented**
GetCapabilities	-
Insert	x
Update	x
Delete	x
GetEntries	x
AcceptChange	-
RejectChange	-
ReviewChanges	-
CreateTopic	-
RemoveTopic	-
ListTopics	-
Subscribe	-
ListSubscriptions	-
PauseSubscription	-
ResumeSubscription	-
CancelSubscription	-
Custom operations	**Implemented**
GetEntryObjectGeometry	x

Table 2: The implemented GSS operations

4.1. GSS Database

The database of the GSS prototype was implemented with the PostgreSQL/PostGIS database. The database is used for storing the information about inserted, updated and deleted features. The database consists of one table where one row represents one change event. The structure of the database is described in *Table 2*.

Database Column	**Data type**	**Description**
GID	integer	Unique identifier for the feature
Type	string	Name of the feature class
Change time	date/time	Timestamp of the change
Operation	string	Operation type (Insert / Update / Delete)
Transaction	string	WFS-T query
Geometry	geometry	Geometry
entryID	integer	Unique identifier for the change event
Author	string	Publisher of the change

Table 2: The structure of the GSS database

The presented table structure enables the database to be queried with various filters. The *GID* column can be used for retrieving all changes that have been made to a single feature. The *Type* column enables querying by feature class. The *Change time* column can be used for performing temporal queries and the *Geometry* column for spatial queries. The *Transaction* column is used for storing the change queries in a WFS-T form (Vretanos 2010). The *entryID* column defines unique identifier for the change events. The *Author* column is used for storing the information about the person who made the change.

4.2. Insert, Update and Delete Operations

The *Insert*, *Update* and *Delete* operations were implemented as HTTP POST operations that read XML data. These operations take in WFS-T queries that are first parsed in the GSS module. After the queries have been parsed, they are sent to the WFS-T service that executes the changes in the master database. After the queries have been executed, the information on the changes is stored into the GSS database. Because the incoming WFS-T Delete operations don't contain the geometries of the features that are to be deleted, their geometries are fetched from the WFS-T service before the Delete operations are executed.

4.3. GetEntries Operation

The *GetEntries* operation was implemented as a HTTP GET operation with a key-value pair (KVP) encoding. The GetEntries operation is used for creating the Atom-formatted replication feed. It supports various query types. The replication feed can be queried with spatial, temporal or id-based filters that allow the subscribers to fetch only those changes that they are interested in. The implemented operation supports also the custom FEATUREID parameter that can be used for retrieving the changes that are related to a specific feature. The implemented parameters of the *GetEntries* operation are listed in *Table 3*.

GetEntries operation parameter	**Implemented**
SERVICE	x
VERSION	x
REQUEST	x
FEED	x
OUTPUTFORMAT	(default format)
STARTPOSITION	-
MAXENTRIES	x
SEARCHTERMS	-
BBOX	x
GEOM	x
SPATIALOP	-
CRS	-
STARTTIME	x
ENDTIME	x
TEMPORALOP	-
ENTRYID	x
FILTER	-
Custom parameters	**Implemented**
FEATUREID	x

Table 3: The parameters of the *GetEntries* operation

4.4. GetEntryObjectGeometry Operation

The *GetEntryObjectGeometry is a custom* operation that was implemented as a HTTP GET operation with KVP encoding. It can be used for retrieving the geometry of a specific change event. It has an entryid parameter that indicates the identifier of an entry whose geometry is to be returned. The operation was included to the GSS prototype because it was found to be useful when the change events are explored in a map application.

5. Discussion

The change proposal reviewing stage was bypassed in the GSS prototype implementation because all changes are seen as authoritative changes that don't need to be accepted. This is called a no-validation workflow (Vretanos 2011b). In many cases, the reviewing phase is essential because the change proposals cannot be trusted without checking their validity. One such scenario is the collection of change proposals via crowd-sourcing.

Also the change proposal reviewing stage could be crowd-sourced. In this scenario, the review process could be implemented so that it doesn't have to rely on a single user making the right reviewing decision. The change proposal might be ultimately accepted only after a certain amount of users have accepted it.

6. Conclusions

The work described in this paper is related to the implementation of a GSS prototype that is used for carrying out an incremental update process between two hydrographic datasets. The experiences gained from the prototype implementation indicate that the GSS seems to be a suitable solution for performing incremental data updates in a web service environment. The broad querying capabilities that the GSS offers to the replication feed enable good control on the update process. The updates can be focused either to a single feature, time interval, geographic area or the whole dataset. The fact, that the GSS is based on open standards makes it suitable for both commercial and open source software.

References

Chormann R, Harrison J (2012) New Hampshire NHD GeoSynchronization Network. Available at:

https://www.fgdc.gov/grants/2010CAP/InterimFinalReports/238-10-2-NH-FinalReport.pdf [Accessed: 2015-04-08).

European Commission (2007) INSPIRE Directive. Available at: http://eur-lex.europa.eu/LexUriServ/ LexUriServ.do?uri=OJ:L:2007:108:0001:0014:EN:PDF [Accessed: 2015-04-08].

Milanović A, Guimet J, Rodellas E, Bolívar M. A. (2010) Interorganizational Geo-Synchronization Using Open Geospatial Consortium's (OGC) Technologies to Share and Harmonize Data in Catalonia, INSPIRE Conference 2010, Krakow. Available at: http://geoportal.cat/geoportal/eng/documents/articles/Ponencia_INSPIRE_2010.pdf [Accessed: 2015-04-08].

Vretanos P. A. (ed.) (2010) OpenGIS Web Feature Service 2.0 Interface Standard. Available at: https://portal.opengeospatial.org/files?artifact_id=39967 [Accessed: 2015-04-08].

Vretanos P. A. (ed.) (2011a) OWS 7 Engineering Report – Geosynchronization service. Available at: *https://portal.opengeospatial.org/files/?artifact_id*=39476 [Accessed: 2015-04-08).

Vretanos, P. A. (ed.) (2011b) OWS-8 - GeoSynchronization Best Practices. Available at: https://portal.opengeospatial.org/files/?artifact_id=46037 [Accessed: 2015-04-08).

Nottingham M, Sayre R (eds.) (2005) The Atom Syndication Format. Available at: http://tools.ietf.org/html/rfc4287 [Accessed: 2015-04-08).

Sparks J, Worrall P, Ehman J, Nail D (2011) Indiana High & Local-Resolution NHD Update Geosynchronization Grant 2010 Category 2: Framework Data Exchange through Automated Geosynchronization. Available at: https://www.fgdc.gov/grants/2010CAP/InterimFinalReports/242-10-2-IN-FinalReport.pdf [Accessed: 2015-04-08).