PROPOSAL FOR A WEB ENCODING SERVICE (WES) FOR SPATIAL DATA TRANSACTION

C. B. Siew [a], S. Peters [a], A. Abdul Rahman [a]

[a] Geospatial Information Infrastructure, Faculty of Geoinformation and Real Estate, Universiti Teknologi Malaysia (UTM), Johor Bahru, Malaysia – bernad@bernadsiew.name, stefanpeters@utm.my, alias@utm.my

KEY WORDS: Encoding, Web service, SDI, Data Transaction

ABSTRACT:

Web services utilizations in Spatial Data Infrastructure (SDI) have been well established and standardized by Open Geospatial Consortium (OGC). Similar web services for 3D SDI are also being established in recent years, with extended capabilities to handle 3D spatial data. The increasing popularity of using *City Geographic Markup Language* (CityGML) for 3D city modelling applications leads to the needs for large spatial data handling for data delivery. This paper revisits the available web services in OGC Web Services (OWS), and propose the background concepts and requirements for encoding spatial data via Web Encoding Service (WES). Furthermore, the paper discusses the data flow of the encoder within web service, e.g. possible integration with Web Processing Service (WPS) or Web 3D Services (W3DS). The integration with available web service could be extended to other available web services for efficient handling of spatial data, especially 3D spatial data.

1. INTRODUCTION

City models are especially important for presentation of business locations and urban development that is required by public and private sectors, such as providing the platform that supports the process from civic participation to decision-making and policy-formulation, as reported by Kolbe et al. (2008). The usages of CityGML or GML in SDI architecture especially web services are quite significant due to the verbosity and declarative nature of XML schema. These benefits also come with drawbacks, i.e. when large file size transaction is expected within client to server, or server to server. Conventional compression method could solve this challenging task only partially, however, not very well handled. For instance, a proper compression workflow for a set of data retrieved from a web service is not available. The approach proposed in OGC best paper (Bruce, 2006) could be used as a generic solution, but a proper spatial data handling service is preferred. Though currently plain CityGML or Geography Mark-up Language (GML) data transaction is being practised across various SDI initiatives, a generalized encoding service could be created for efficient data handling. CityGML is based on GML which is an XML standard – a popular W3C standard in data exchange and sharing usage in computing systems due to its readable and self-describing benefits. However, given the cost involved in storing and transferring the schema over distributed networks, transmitting models in CityGML format is considered impractical, especially when the city model is used for visualization purposes over the distributed environment. The use of X3D for city model representation in Web 3D Services (W3DS) (Schilling and Kolbe, 2010) is then utilized. On the other hand, dealing with semantic data that is required by analysis for orchestration in Web Processing Service (WPS), a data model with semantics is necessary. To deal with these requirements, transmitting CityGML in large datasets should be well-handled.

Encoding as a service for spatial data gains binary transaction benefit and maintain interoperability. Such service mimic the implementation of WPS that could be used for web service chaining. This service could then be used as an intermediate service specific for spatial encoding. This web service interface inherits the OWS standard interface, which allows expandable and customizable encoding capability for spatial data (XML-schema). Specific encoding scheme could be a step forward towards establishing an application network protocol for efficient and manageable data transaction.

In this paper, Section 2 revisits various OWS and list down its current usages. Section 3 discusses the concept diagram of the proposed WES while Section 4 describes the data flow of the WES in various scenarios. Section 5 then discusses the outlook of the web service and potential integration with other existing web services.

2. BACKGROUND

Previous local SDI frameworks are built on web services mainly from OWS. Galip (2007) showcased a SDI framework (Figure 1) for real time data grid utilizing binary XML as middleware data transaction. This example showed the requirements for having GIS over distributed environment; compare to existing desktop GIS solutions. In Gong et. al (2009), framework for geospatial information dissemination on the web (see Figure 2) include encoding as a service as a component in data dissemination.

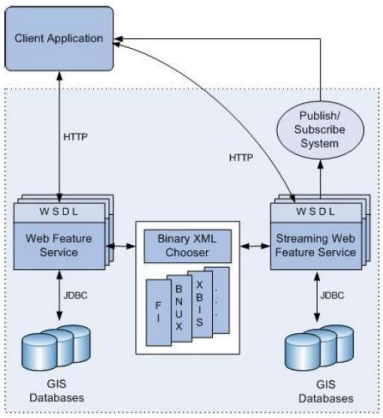

Figure 1. The SDI framework supporting real time data grid incorporating binary XML framework (Galip, 2007).

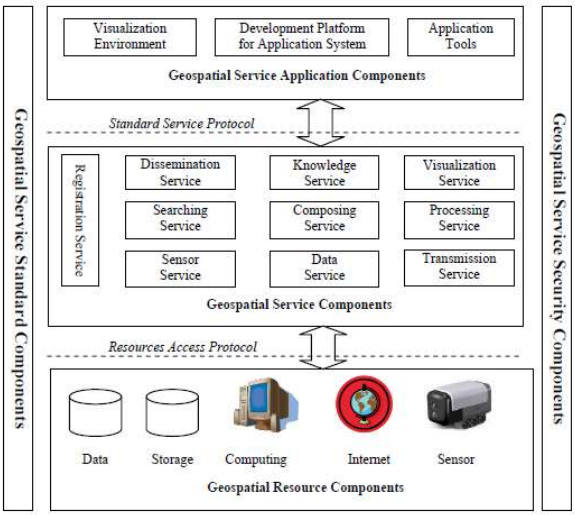

Figure 2. Geospatial services components (Gong et. al, 2009).

Currently OWS is being investigated for 3D spatial data include Web Feature Service (WFS) (Vretanos, 2010), Web Processing Service (WPS) (Schut, 2007) and Web 3D Service (W3DS) (Schilling and Kolbe, 2010). All web services are using HTTP protocol. For WFS, the main usage is facilitating retrieval and updating of geospatial data which is encoded in Geographical Markup Language (GML). The data delivery is retrieved or updated via GML format. Operation processes such as *GetGMLObject*, or *GetFeature* is the interface to fetch elements' instances.

Besides, WPS facilitates geospatial processes and publishing. Processes are defined as any involvement of calculation and algorithms or models that operates on spatial-referenced data while publishing is defined as enabling machine-readable binding information and human-readable metadata for discovery and use. Discovery and binding of information is done by client. W3DS, on the other hand, is a service to present 3D portrayal data delivering X3D as the format for graphical representation. W3DS offers 3D layer on top of WFS layer for visualization. As all the web services is compliant to OWS, the generalized interface for service discovery and execution is categorized into 3 main operations: 1) *GetCapabilities*, 2) *DescribeProcess* (Custom Operation) and 3) *Execute*.

The *GetCapabilities* operation allows client to request and receive service metadata that describe the abilities of the server implementation. It provides naming and the descriptions of the processes offered by WPS instance. The operation implemented in OWS supports negotiation of the specification version being used for client-server interactions. Besides that, *DescribeProcess* allows client to request and receive back information detail about the processes that could be executed on the service instance. This includes the required input, permitted formatting, and the output that will be produced whereas *Execute* operation allows client to run a specified process implemented by the web service, by using required input parameters and return the output produced (Schut, 2007).

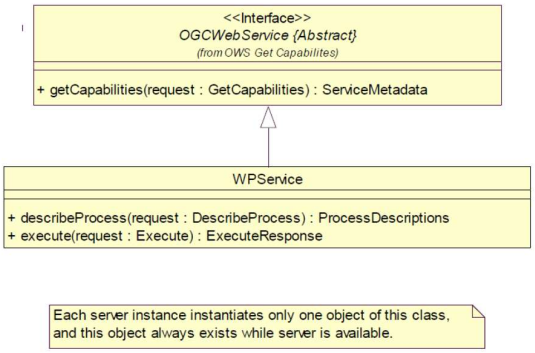

Figure 3. The WPS operations (Schut, 2007).

In Binary XML document (Bruce, 2006) shows how the encoding process could be done for the XML data. The approach could be further extended for web service implementation via standard OWS operations. On the contrary, GeoPackage (Daisey, 2014) from OGC standard discussed on limitation of small devices in terms of storage and network connectivity, and how GeoPackage could improve data dissemination for these environments. GeoPackage emphasises on SQLite usage and data types for SQLite DB. This approach is seen useful in terms of sending the data to client in a package, however, the relationship and workflow with other web services are not discussed.

3. WES INTERFACE

Similar to OWS, WES employs the generalize workflow as depicted in Figure 4.

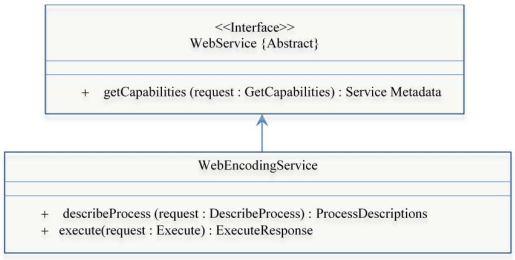

Figure 4. The WES Interface based on OGC.

Considering a simple scenario where a request chains this Web Encoding Service as middleware to obtain encoded data, the response of this web service to a *GetCapabilities* request indicates that this web service supports the operation of "*encodeGML*", where this operation limits to encode only one XML input at an instance. The response to a *DescribeProcess* request for the "*encodeGML*" process might indicate that it requires one input which is "an XML" and this input must be provided in GML 2.2. Furthermore, the process will produce one output, in a ZIP archive that contains several containers with information in it and delivered as a web-accessible resource. The client would then run the process by calling *Execute* operation, and then provide the input of the data embedded directly within the request (stream), and identify that the output should be stored as a web-accessible resource.

The nature of the web encoding service is similar to WPS, which is a middleware service. Middleware service obtains data from an external resource in order to run a process on the local implementation. As a middleware service, it allows current software interfaces to be wrapped up and presented to the

network as web services. This proposed web service wraps encoding capabilities and the format of the encoding service is based on UTF-8. This is due to the fact that UTF-8 and UTF-16 are widely accepted in browser-based client.

For spatial data handling in WES, for example, binary encoding of X3D could be implemented as encodeX3D identifier. Its internal operation depends on the further implementation of this WES in future. Currently, CitySAC (Siew and AbdulRahman, 2013) could be used in this WES as the internal encoding engine for encodeGML operation.

4. WES DATA FLOW AND OPERATIONS

In *GetCapabilities* operation, the request parameters are shown in Table 1. Table 2 shows the response of the service metadata (XML) for discovery purpose. Table 3 shows the request parameters of *DescribeProcess* operation, while Table 4 shows the response of *DescribeProcess* operation. On the other hand, Table 5 denotes the *Execute* process with its example query over URL GET request.

URL (GET request)	http://foo.bar/foo?service=WES&Request=GetCapabilities&AcceptVersions=0.1.0&language=en		
Name	Multiplicity	Client Implementation	Server Implementation
Service	One (mandatory)	Parameter shall be implemented by all clients, using specific value	Parameter shall be implemented by all servers via checking of specific value upon receiving each parameter
Request	One (mandatory)	Parameter shall be implemented by all clients, using specific value	Parameter shall be implemented by all servers via checking of specific value upon receiving each parameter
AcceptVersions	Zero or one (optional)	Optional implementation by all clients	Parameter shall be implemented by all servers via checking of specific value upon receiving each parameter
language	Zero or one (optional)	Optional implementation by all clients	Parameter shall be implemented by all servers via checking of specific value upon receiving each parameter

Table 1. The request parameters in *GetCapabilities* WES.

Description	The response of service metadata		
Name	Definition	Data type and value	Multiplicity
Service	Service Identifier	Character String type e.g. "WES"	One (mandatory)
Lang	Language identifier	Character String type	One (mandatory)
ProcessOfferings	Brief descriptions of the processes offered by the server without inputs or outputs	ProcessBrief data structure	One (mandatory)
Languages	Languages supported	Languages data structure	One (mandatory)

Table 2. The response of WES *GetCapabilities*.

The response from the *GetCapabilities* operation will be as shown in Table 2. The ProcessOfferings refer to the varieties of processes allowed in WES. The structure of the *ProcessOfferings* return will follow *ProcessBrief* structure. In Table 3, the *DescribeProcess* sends the request to server to notify the operation that should be executed by providing the operation in the identifier parameter. The return of the *DescribeProcess* would be the detail of the operation as indicated in the *ProcessIdentifier* as well as the version of the process as shown in Table 4.

Description	**The request of the operation**		
Example	http://foo.bar/foo?Service=WES&Request=DescribeProcess&Version=0.1.0&Identifier=encodeGML		
Name	**Definition**	**Data type and value**	**Multiplicity**
service	Service Identifier	Character String type e.g. "WES"	One (mandatory)
request	Operation name	Character String type Value is "DescribeProcess"	One (mandatory)
version	Version for operation	Character String type, Specified value indicates the schema version. Value is "0.1.0"	One (mandatory)
Identifier	Process Identifier	Character String type, Value is process identifier defined in ProcessOfferings	One or more (mandatory)

Table 3: The request of WES *DescribeProcess*.

Description	The Process Description detail		
Name	**Definition**	**Data type and value**	**Multiplicity**
Identifier	Identifier from ProcessBrief data structure	Character String type	One (mandatory)
Title	Inherit from ProcessBrief data structure	Character String type	One (mandatory)
processVersion	The specific process version for specific application	Character String type, Specified value indicates the schema version.	One (mandatory)
DataInputs	List of inputs to this process	The GML or CityGML according to accepted version	One (mandatory)
ProcessOutputs	List of outputs from this process	Encoded data in archival (ZIP mimetype)	One (mandatory)

Table 4: The response of *DescribeProcess*.

Table 5 shows the request of the operation *Execute* and the inputs parameter is mandatory and the output of the execution is also indicated in this request string. In this request, the *GET* input generated from WPS or WFS or other web services could be used as input in this stage, then the output of the encoded data could be further defined in the chaining of the web services.

In this case, the identifier of the operation would be "*encodeGML*" and this interface will perform encoding with the data return type defined by the *RawDataOutput*. Within this "*encodeGML*", the encoding is currently implementing CitySAC. CitySAC is using dictionary encoding, all occurrences are indexed and stored. Furthermore, geometries are built in chunk of 65,000 face sets where each face represents a polygon in CityGML. The main idea of the encoder is to follow the XML original structure so query-able capability is retained and compressed content is retrievable. The encoder defines each representation as a symbol denoted in 16-bits. The encoding flow is depicted in Figure 5.

Description	The Execute operation request parameter		
Example	http://foo.bar/foo?request=Execute&service=WES&version=0.1.0&language=en&Identifier=encodeGML&DataInputs=http%3A%2F%2Ffoo.bar%2Ffoo%2Finput.gml&RawDataOutput=EncodedBuildings@Format=application/octet-stream		
Name	**Definition**	**Data type and value**	**Multiplicity**
Service	Service type identifier	Character String type Value is "WES"	One (mandatory)
Request	Operation name	Character String type Value is the operation name e.g. "Execute"	One (mandatory)
version	The specific process version for specific operation	Character String type, Specified value indicates the schema version.	One (mandatory)
Identifier	Unique identifier for the name of the process	Character String type. Value is the process identifier used in Capabilities document	One (mandatory)
DataInputs	XML inputs for encoding purposes	GML or CityGML or XML compliant documents	One (mandatory)
RawDataOutput	Raw output return from the web service	Raw data compliant to the mimeType	One (mandatory)

Table 5: The request of *Execute* operation.

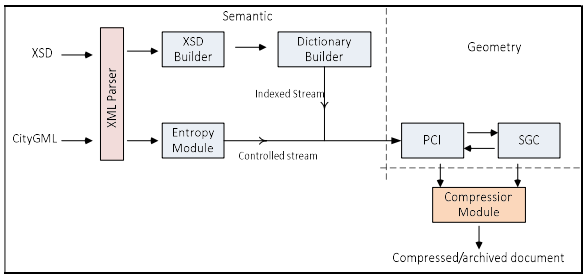

Figure 5. The CitySAC encoding workflow in *encodeGML* operation (Siew and Abdul Rahman, 2013).

Encoded data will be decoded by calling on-demand script library in other web services, as the library will decode based on the dictionary built on-the-fly. The decoding process could be depicted as Figure 6. Note that this process is just a reference for decoding a simple 3D spatial object from a scene, queried and encoded by WES. The input of the decoding process should be the encoded output produced by "*encodeGML*" operation in form of a package/archive.

The entire web service workflow and usage for encoding spatial data is similar to the intermediary service in the WPS nature. This encoding as a service concept could then be realized in the chaining of web services in OGC. The decoding process could be done in client side, or in web services that required further usages on the encoded data. Decoding process is similar for both, client side for thin client, as well as for thick client due to the code on demand advantage in the decoding requirement.

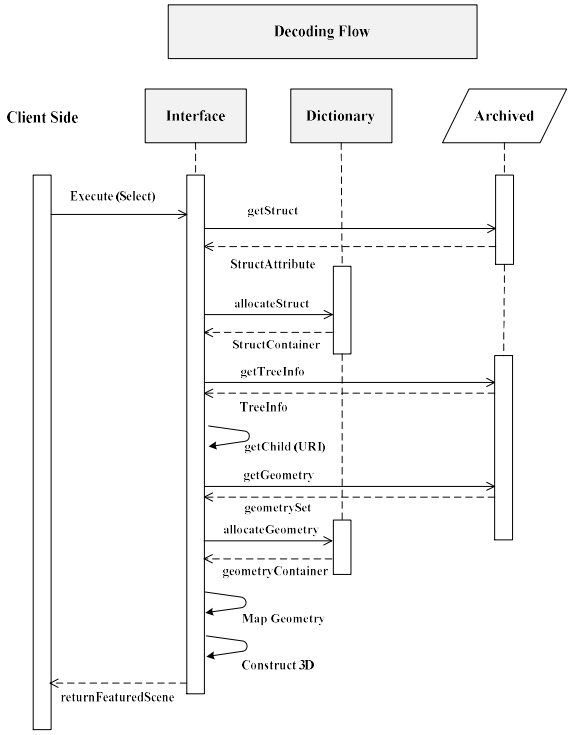

Figure 6. The decoding process flow example.

Figure 6 depicts a brief example for decoding data at thin client. Decoding interface is written in browser scripting language, and dictionaries of the encoded data are archived in LZMA compression scheme. The decoding process starts with a simple *SELECT* of a building, then the *getStruct* method will obtain the document structure from the archive, by returning *StructAttribute*. Structures are allocated in *allocateStruct* and the container of the structures will be filled and return to the main operation. Then, *getTreeInfo* method will retrieve entire *TreeInfo* from the archive, therefore parent and child attributes could be used in *getChild* method with given URI. Next, the geometries will be obtained from the archive *getGeometry* and the set of geometry would be retrieved and allocated using *allocateGeometry* method. Geometry container is now filled with geometries and mapped into the parent and child tree, and then 3D buildings are constructed and returned as *FeaturedScene*.

5. CONCLUDING REMARKS

In this paper we showcase the draft of a new web service called WES for spatial data compression handling. The proposed WES is a web service to handle binary compression of spatial data in distributed environment, as its objective is to realise encoding as a service compliant to OWS standards. Initially some OWS are reviewed and discussed in the background section, where the related OWS for binary transaction operations are discussed. We have shown the nature of WPS as a middleware, which mechanism could be useful for WES. We also reviewed BXML best paper document and discovered the requirements to create a web service specified for compression usage. The operations of the WES that is similar to those OWS operations are also demonstrated. Example of how spatial data is encoded in the web service operation is also presented, along with the decoding process. This web service is a step towards a network protocol embedding the binary schema similar to the concept of the WES.

With WES, specific encoding could be employed and used over web service chaining and orchestration. Various existing or new web service could be easily chained with WES. The WES could be further extended by employing more encoding scheme in the *ProcessOffering*.

ACKNOWLEDGEMENTS

We would like to convey our deepest acknowledgement to Ministry of Higher Education (MOHE), Malaysia for the scholarship under the program MyPhD - MyBrain15 and enabling us to carry out this research project.

REFERENCES

Bruce C., 2006, Binary Extensible Markup Language (BXML) Encoding Specification, OGC Best Practices document, OGC® 03-002r9

Daisey Paul, 2014, OGC® GeoPackage Encoding Standard v1.0, OGC Standard Document, OGC 12-128r10

Galip Aydin, 2007, Service Oriented Architecture for Geographic Information Sys-tems Supporting Real Time Data Grids, PhD Dissertation, Department of Computer Science Indiana University

Gong Jianya, Huayi Wu, Wenxiu Gao, Peng Yue, Xinyan Zhu, 2009, Geospatial Service Web, Geospatial Technology for Earth Observation, Springer Science, DOI 10.1007/978-1-4419-0050-0_13.

Kolbe T. H., Gerhard K., Nagel C., Stadler A., 2008, 3D Geodatabase Berlin ver-sion 2.0.1a, Fachgebiet Methodik der Geoinformationstechnik der TU Berlin

Panagiotis A. Vretanos, 2010, OpenGIS Web Feature Service 2.0 Interface Standard, OGC Implementation Standard, OGC 09-025r1

Schilling A. Kolbe T.H., 2010. Draft for Candidate OpenGIS® Web 3D Service Interface Standard, Version: 0.4.0, OGC 09-104r1 pp. 95.

Schut P., 2007, OpenGIS® Web Processing Service, OpenGIS Standard, OGC 05-007r7

Siew, C. Bernad, Alias A Rahman, 2013, A schema-aware encoder for Putrajaya 3d, Urban and Regional Data Management, UDMS Annual 2013 - Proceedings of the Urban Data Management Society Symposium 2013 , pp. 181-190.