

# Open Geospatial Consortium Inc.

Date: 2007-05-10

Reference number of this OGC® IP initiative document: **OGC 06-182r1**

Version: **0.9.1**

Category: OGC® Discussion Paper

Editor: Steven Keens

## OWS-4 WPS IPR

### Discussions, findings, and use of WPS in OWS-4

#### Copyright notice

Copyright © 2007 Open Geospatial Consortium, Inc. All Rights Reserved.  
To obtain additional rights of use, visit <http://www.opengeospatial.org/legal/>.

#### Warning

This document is not an OGC Standard. This document is an OGC Discussion Paper and is therefore 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, an OGC Discussion Paper should not be referenced as required or mandatory technology in procurements.

Document type: Discussion Paper  
Document subtype: if applicable  
Document stage: Draft  
Document language: English

## Contents

i.	Preface.....	vi
ii.	Submitting organizations .....	vi
iii.	Document contributor contact points .....	vi
iv.	Revision history.....	vii
	Foreword.....	viii
	Introduction.....	ix
1	Scope.....	1
2	Terms and definitions .....	1
3	Conventions .....	2
3.1	Symbols (and abbreviated terms).....	2
3.2	UML notation .....	2
3.3	Document terms and definitions.....	2
4	Recommendations .....	2
4.1.1	Process definition .....	2
4.1.2	Geo-process definition .....	3
4.1.3	Relation between WPS and SOAP/WSDL .....	3
4.2	Enhancements .....	3
4.2.1	Push(ing) execute request results .....	3
4.2.2	Change DescribeProcess response to return an XML Schema .....	4
4.2.3	Add Resource Management Interface like in WSRF .....	4
4.2.4	Miscellaneous subjects.....	7
5	OWS-4 WPS Processes .....	8
5.1	Overview .....	8
5.2	Generalization Process .....	8
5.2.1	Enhanced GPW Scenario.....	8
5.2.2	WPS for generalization.....	9
5.2.3	An abstract interface for data reduction .....	9
5.2.4	Implementation .....	10
5.2.5	Simplification Results .....	11
5.3	Clipping Process.....	11
5.3.1	Introduction.....	11
5.3.2	Operations .....	12
5.3.3	WFS-T-based WPS-Clipping.....	14
5.3.4	Service Demo using BPELPower: .....	17
5.4	Binary Grid Process.....	18
5.5	Feature Fusion Process.....	19
5.5.1	Requirements.....	19
5.5.2	Schema Mapping and Assembly Tool (SMAT).....	20
5.5.3	Feature Collection Transformation and Feature Collection Appending Services.....	20

5.5.4	XML Joins and Unions.....	21
5.5.5	Data models and "governance" .....	21
5.5.6	Questions.....	22
6	WPS, WSDL, and SOAP .....	22
6.1	Introduction.....	22
6.2	Web enabling a geo-process .....	23
6.2.1	SOAP & WSDL in the Binary Grid Processing Service .....	24
6.3	WPS WSDL document style .....	25
6.3.1	WPS publishes one all-encompassing WSDL document.....	25
6.3.2	WPS publishes several WSDL documents.....	25
6.3.3	WSDL and the OGC Capabilities .....	25
6.4	WSDL location URL.....	25
6.4.1	Overview .....	25
6.4.2	WSDL location URL construction mechanisms.....	26
6.4.3	Fully formed WSDL location URL .....	27
6.5	Requires further investigation .....	27
6.6	SOAP .....	27
Annex A	Workflows.....	28
A.1	GPW Workflow Summary.....	28
A.1.1	Data reduction workflow.....	28
A.1.2	Synchronous feature update workflow .....	29
A.1.3	Asynchronous feature update workflow.....	30
A.1.4	Issues 31	
Annex B	Change Requests .....	32
B.1	Change request - Support POST in ComplexValueReference element.....	32
Annex C	WSDL Examples .....	33
C.1	GMU WPS WSDL .....	33
C.2	Binary Grid Processing Service WSDL .....	35
Annex D	WPS best practice implementation .....	42
D.1	Architecture overview .....	42
Annex E	WPS XML examples.....	43
E.1	WPS DescribeProcess example for Douglas Peucker simplification.....	43
Annex F	Annex F OWS-4 Earth Observation Demo .....	45
	Bibliography .....	47

<b>Figures</b>	<b>Page</b>
<b>Figure 1: Overlay of the original and simplified MSD3 road data set with a data reduction of 40 %.</b> .....	<b>11</b>
<b>Figure 2: Fuzzy clipping</b> .....	<b>15</b>
<b>Figure 3: Before Clipping</b> .....	<b>16</b>
<b>Figure 4: After clipping</b> .....	<b>17</b>
<b>Figure 5: DataFed service chain</b> .....	<b>19</b>
<b>Figure 6: Web enabling a geo-process</b> .....	<b>23</b>
<b>Figure 8: Data reduction workflow sequence</b> .....	<b>28</b>
<b>Figure 9: Synchronous feature update workflow</b> .....	<b>29</b>
<b>Figure 10: Feature update workflow</b> .....	<b>30</b>
<b>Figure 11: Pluggable architecture for WPS</b> .....	<b>42</b>
<b>Figure 12: Components and services in the OWS-4 Earth Observation demo</b> .....	<b>46</b>

<b>Tables</b>	<b>Page</b>
<b>Table 1: Generalization parameters .....</b>	<b>10</b>
<b>Table 2: Clipping examples .....</b>	<b>12</b>
<b>Table 3: Clipping implementation links .....</b>	<b>17</b>
<b>Table 4: Binary grid process parameters .....</b>	<b>18</b>
<b>Table 5: Data reduction workflow .....</b>	<b>28</b>
<b>Table 6: Synchronous feature update workflow.....</b>	<b>29</b>
<b>Table 7: Asynchronous feature update workflow .....</b>	<b>30</b>

**i. Preface**

Suggested additions, changes, and comments on this draft report are welcome and encouraged. Such suggestions may be submitted by OGC portal message, email message, or by making suggested changes in an edited copy of this document.

The changes made in this document version, relative to the previous version, are tracked by Microsoft Word, and can be viewed if desired. If you choose to submit suggested changes by editing this document, please first accept all the current changes, and then make your suggested changes with change tracking on.

**ii. Submitting organizations**

The following organizations submitted this document to the Open Geospatial Consortium Inc.

PCI Geomatics

**iii. Document contributor contact points**

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

Contact	Company	Email
Steven Keens	PCI Geomatics	Skeens [at] pcigeomatics.com

## iv. Revision history

Date	Release	Editor	Primary clauses modified	Description
September 18, 2006	0.0.1	Steven Keens	All	Initial revision.
November 29, 2006	1.0.0	Steven Keens	Most	Final draft.
December 7, 2006	1.0.0	Theodor Foerster	Generalization, WPS best practice implementation	Some WPS related revision
December 8, 2006	1.0.0	Stefan Falke	Binary grid processing	
December 8, 2006	1.0.0	Steven Keens	5	Added enhancements subclause.
December 9, 2006	1.0.0	Theodor Foerster	5	Minor changes
December 22, 2006	1.0.0	Theodor Foerster	All	Minor review
January 5, 2007	1.0.1	Steven Keens	Many	Incorporated comments and edits from Peisheng Zhao, Stefan Falke, Theodor Foerster.
2007-5-10	0.9.1	Carl Reed	Various	Preparation for posting as a DP

## Foreword

This work is an interoperability report compiled by the Geo-Processing Workflow group (GPW) of the OGC Web Services, Phase 4 initiative of the OGC Interoperability Program.

This document presents the work completed with respect to the WPS.

This is not a normative document.

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 Inc. 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 specification set forth in this document, and to provide supporting documentation.



## **Introduction**

OWS-4 has as one of its mandates the requirement to use the WPS in its workflows. This document reviews the material discussed during the OWS-4 project, describes the WPS processes deployed in the workflows, and offers suggestions to the OGC to move forward with the WPS. The GPW thread also discussed one change request that is submitted with this document and to the OGC.



# OWS-4 WPS IPR

## 1 Scope

This document reviews the material discussed during the OWS-4 project, describes the WPS processes deployed in the workflows, and offers suggestions to the OGC to move forward with the WPS. The GPW thread also discussed one change request that is submitted with this document and to the OGC.

The original goal for this document was to write a profile IPR. In the end this document does not document any WPS profiles. That is because the OWS-4 GPW WPS participants never really discussed profiling the WPS. Due to the nature of the WPS, it is easy to forego considering profiles and simply use the existing capabilities described in the WPS specification. Thus, instead of writing one or more profiles for the WPS, this document discusses the WPS processes that were implemented, amongst other things. Those processes could be made into WPS profiles if there is need.

## 2 Terms and definitions

For the purposes of this specification, the definitions given in [1], [2], and [3] shall apply.

In addition, the following terms and definitions apply:

### 3.1

#### **service chain**

sequence of services where, for each adjacent pair of services, occurrence of the first action is necessary for the occurrence of the second action

### 3.2

#### **workflow**

automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules

### 3 Conventions

#### 3.1 Symbols (and abbreviated terms)

This document uses the symbols and abbreviations defined in Subclause 5.1 of the OGC Web Services Common Implementation Specification 1.1.0 draft [OGC 06-121].

OWS-4	OGC Web Services, Phase 4
GPW	GeoProcessing Workflow
WPS	Web Processes Service
WSDL	Web Service Definition Language
SOAP	Simple Object Access Protocol
BPEL	Business Process Execution Language
REST	Representational State Transfer

#### 3.2 UML notation

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

#### 3.3 Document terms and definitions

This document uses the specification terms defined in Subclause 5.3 of [OGC 06-121].

## 4 Recommendations

### 4.1.1 Process definition

This recommendation to the OGC is to help define the terms “process” and a “service”. Naturally, people often use and intermix the terms “process” and “service” because of their similar meanings. In light of that, it is reasonable to have the OGC clarify the distinction between the two terms.

The problem goes further than just the distinction between “process” and “service”. The issue arises again when considering the different definitions of the same term in various other specifications. These concepts are all similar in that they describe a process, process chain, or the lineage. The difference is only the point of views.

Specification	Specification term	Description	Point of view
SensorML	Process	On-demand processing of Observations - Process chains for geolocation or higher-	Process chain $P^1-P^P$ produces results $R^1-R^N$ from input $I^1-I^i$

		level processing and executed on-demand without a priori knowledge of the sensor or processor characteristics.	
WPS	Process	Model or calculation that is made available at a service instance. This is not a chain - unless it is an opaque chain.	Process $P^1$ produces results $R^1-R^N$ from input $I^1-I^i$
ISO 19115	Lineage and process step	The history or list of process steps performed to create the data described by the ISO document.	Result $R^1$ was produced by process chain $P^1-P^p$ with inputs $I^1-I^i$
BPEL		Closer to a programming language but something that confuses the issue.	

The OGC needs to clarify the distinctions between these concepts, determine the unifying concepts, and perhaps find an abstract term to classify all of these concepts under one umbrella.

#### 4.1.2 Geo-process definition

The question arose whether the OGC should emphasize a WPS process as being a “geoprocess” – operations that work on the spatial and/or temporal dimensions of geospatial data or any type of data. It is likely that most WPS processes will work on only spatial-temporal data. However, the WPS permits any type of process and thus we can not limit it.

It is recommended that the term geoprocess be defined as a process that works on the spatial-temporal dimensions of geospatial data. It is also recommended that the WPS specification clearly indicate that the WPS is designed meant to work on geoprocesses.

#### 4.1.3 Relation between WPS and SOAP/WSDL

The relationship between the WPS and SOAP/WSDL was always a point of discussion within our the OWS-4 GPW WPS teleconferences. That subject is discussed in clause WPS, WSDL, and SOAP6.

### 4.2 Enhancements

Several enhancements to the WPS were discussed during the OWS-4 project. Only one of those eventually became a change request (see [5]). This section examines the other enhancements discussed.

#### 4.2.1 Push(ing) execute request results

Currently, the WPS stores the process results for later retrieval by the client on a web-accessible source as a flat file or returns the process results immediately. This design

prevents the client from stating where to the WPS should put the results. Modifying the WPS to support push would save some server resources and could save some bandwidth.

If, or when, pushing results is considered the WPS should be designed such that it can push the data via transactions to other OGC services. For example, a WPS process producing GML features could conceivably push those features to a WFS-T via a transaction. Currently, that is not possible because the WPS wasn't designed with that use case in mind. Thus, the WPS specification needs to be designed to permit pushing data to other OGC services (also introduced as smart storing). This approach is beneficial, as it provides clients a way to query and retrieve the process results in an individual way. Imagine a complex feature or raster based process, which produces gigabytes of data. Such a process result could then be served through a standardized interface and could thus become accessible to lots of (standard) OGC web clients. It eases the access to these process results in two ways: computationally (retrieve) and logically (query). However, the design should be generic and not cater to specific OGC services. How to push data to a WCS or WFS via a transaction could be written as a third document binding the WPS to the WCS or WFS. It could be done in a profile, a best practices document, or it could be added directly as an optional part in the WPS specification.

#### **4.2.2 Change DescribeProcess response to return an XML Schema**

Currently, the DescribeProcessResponse is used to provide detailed information about parameters of the process plus some additional metadata (abstract...). During the OWS-4 discussions related to WSDL adoption in the WPS, it turned out that the xml-encoding of these processes in the DescribeProcessResponse is not sufficient. However it is not possible to link the DescribeProcessResponse directly in the WSDL document. But changing the encoding to XML schema would enable the direct embedding of the ProcessDescription in the WSDL document. As it seems, the current notion of the DescribeProcessResponse follows the ideas from the WebCoverageService. However it has to be admitted, that using xml schemas would provide a more normative view on processes. Using schemata for querying encoding is also used in DescribeFeatureType of WFS. So using XML schema for DescribeProcessResponse would improve WPS in such ways as:

- Enable validation of Execute requests
- direct embedding in WSDL
- Enable to build application profiles for processes such as for GML

#### **4.2.3 Add Resource Management Interface like in WSRF**

The WSRF (WS Resource Framework) discusses a technique for web services to support statefull resource management (see [14]). The WPS should consider adding WSRF like capabilities to the specification

The specification defines new operations that allow clients to manage resources. This type of management is necessary to make processing large data viable. See the presentation available at [http://portal.opengeospatial.org/files/?artifact\\_id=15754](http://portal.opengeospatial.org/files/?artifact_id=15754) to help understand.

Add the following new resource management operations to the WPS:

- **PutResource**: used to send a resource to the WPS. The response contains a resource id and a termination time. The resource Id is used within its other requests. It can be used within the Execute requests as the input parameter value to a process.
- **GetResource**: used to retrieve a copy of the resource. This method is already defined by OWS Common as `GetResourceById`. It will be used most often by the client to retrieve the output from a process.
- **DestroyResource**: used to destroy the specified resource from the WPS.
- **SetTerminationTime** is used to change when the resource should be destroyed automatically by the server.

All of the operations provide access to what is called the **Temporary Resource Store (TRS)**.

#### 4.2.3.1 PutResource Operation

The `PutResource` operation is used to send a resource to the WPS and puts it in the TRS. This method should be used when the resource is likely to be used by several processes.

The old WPS method of doing things (ie, using a reference to a remote resource for input) required that the client make the resource available on another server, where the WPS could fetch it. Now, the client can just push the data up to the WPS. It also helps circumvent firewall issues.

It is a dangerous operation because it can be used to overload the server's TRS.

#### *Questions*

- Should the client state the MIME type when putting a resource into a WPS's TRS?
- Should the server publish a limit on the size of resource it will accept?
- Should the server publish its default `TerminationTime`?
- Should the client be able to suggest an initial `TerminationTime` in a `PutResource`?

- Is there any other type of metadata that we might want to add the request?

The PutResource response contains a ResourceId and a TerminationTime. The ResourceId is used within other requests to reference the resource. For example, the ResourceId can be used within Execute requests as the input parameter value to a process. The TerminationTime tells the client when the resource will no longer be available, or rather, destroyed.

#### *Questions*

- What is the format or type of the ResourceId? Likely this should be an OWS Identifier.
- What time format will we use for the TerminationTime.

#### **4.2.3.2 GetResource Operation**

The GetResource operation is used to get the resource identified. This method is already defined by OWS Common as GetResourceById. It will be used most often by the client to retrieve the output from an Execute request to a process.

The GetResource operation shall return an exact copy of the resource identified.

#### **4.2.3.3 Resource Destruction**

There are two ways for a resource to be destroyed: explicitly with the DestroyResource operation and automatically by the server at or after the TerminationTime.

When a resource is destroyed it is no longer available or accessible in any way. The resource id is now meaningless and will cause exceptions if used during any attempts to access the resource. Effectively, this means it was deleted from the database, disk drive, or whatever form of storage used. Practically, it is really up to the server if it wants to delete the resource or not.

The DestroyResource operation is used to destroy the specified resource from the WPS. Practically, this is really just a suggestion to the server to destroy the resource. The server is free to do what it wants to the resource. The server must still maintain the semantics of what it means for a resource to be destroyed.

#### **4.2.3.4 SetTerminationTime Operation**

Allows a client to request that the TerminationTime be changed, either to extend it or shorten it. It is up to the server to decide if it will do so. The response will indicate if it passed or failed. If the server accepts then the server shall not destroy the resource anytime before the TerminationTime. Only under very exceptional conditions can the resource be destroyed before the TerminationTime. The resource will be made unavailable to a client after the termination time since it is considered to be destroyed.



#### 4.2.4 Miscellaneous subjects

The following subjects were mentioned but not delved into:

- Asynchronous communication - for long term processes by utilizing WNS.
- Mobile code – transmitting the process rather than the data.

## **5 OWS-4 WPS Processes**

### **5.1 Overview**

The OWS-4 project introduced three geospatial services that were implemented as WPS processes:

- Generalization
- Clipping
- Binary Grid Processing Service

Two other services were not implemented as WPS processes but could have been, namely:

- Feature Fusion Service (FFS)
- Topological Quality Assessment Service (TQAS)

### **5.2 Generalization Process**

Generalization is about extracting spatial data according to purpose and scale. Generalization can be divided into model and cartographic generalization. Model generalization describes the transformation of spatial data from source model to a target model. The target model incorporates a certain level of detail (LoD) (semantic and geometric). Cartographic generalization deals with the representation of the data and tries to solve graphical conflicts.

One purpose of generalization is to reduce data (lower the level of detail in geometrical sense). There exist different generalization operators to simplify geometries and amalgamate objects.

#### **5.2.1 Enhanced GPW Scenario**

The clip-zip-ship use case's main purpose is to perform data reduction so that a mobile client with limited bandwidth is not overloaded. Generalization could be included in that use case and perform further data reduction. The clip-zip-ship service does not reduce the data inside the spatial extent. As we talk about data, which is only used for display purposes, the rate of the transferred data and the actual displayed data is quite low, due to the limited display capabilities of mobile devices. This rate decreases during scale change (zoom out) even more. Generalization could help to maintain this ratio, by eliminating at least the aspects of data, which share the same pixel on the display. Thus generalization could improve the network load caused by transferring display data. Additionally as some features are not able to be perceived as distinct objects, due to scale change, these objects could amalgamate. This would also improve the ratio of displayed and transferred data.

All generalization operators result in data elimination. This elimination is done in different ways, depending on the geometry type, the spatial context and the granularity of the objects. Thus, for this project the following operators were suggested:

- Simplification (deleting unnecessary points of a line geometry), most applicable in this context, easy to implement.
- Amalgamation (merging the geometries of different features to a new geometry of the same geometry type)
- Elimination (deleting unperceivable objects)

Other possible operators, which aim more at data modelling, but also could result in data reduction

- Collapse (changing the geometry type by decreasing the dimension of the geometry)
- Combine (merging different feature to new geometry type of higher order, increase dimension of geometry)

### 5.2.2 WPS for generalization

Generalization is carried out by complex geo-processes based on generalization operators. Due to the evolving WPS standard and the application of web-based generalization the WPS provides a sufficient environment to publish these operators.

### 5.2.3 An abstract interface for data reduction

Simplification and amalgamation are applied for data reduction. However both operators are quite different in their semantics:

- Simplification reduces the data elements (points) within a line, thus a line, will always be a line, but simplified
- Amalgamation reduces the features by combining aligned geometries to a new one. Thus geometry objects will be deleted and replaced by composed ones.

Additionally simplification and amalgamation always depend on the characteristics of the current data set, thus an interface has to reflect this uniqueness. Therefore I suggest having an abstract interface, which includes a relative parameter for the generalization operator. The idea is to indicate only the percentage of data reduction. The amalgamation and the simplification algorithm can calculate an appropriate data set. Indicating a percentage of data reduction is more meaningful than just a simple tolerance value, which actually involves some experience with the data set and the algorithm itself. The percentage indicates:

- Simplification (a.k.a. line simplification): how many data points of the original lines are included in the source data set
- Amalgamation: how many polygons are transferred

Thus the abstract interface includes the following parameters:

- The source dataset
- The percentage of data reduction

#### 5.2.4 Implementation

Due to lack of time the only generalization process implemented and used within the data reduction workflow was Simplification. It uses the Douglas Peucker Simplification Algorithm.

The proposed abstract interface as described in 5.2.3 will not build the basis for the implementation, as it goes along with significant impact on processing performance.

The implementation of the generalization process is based on the 52north implementation of the WPS:

<https://incubator.52north.org/twiki/bin/view/Processing/52nWebProcessingService>.

**Table 1: Generalization parameters**

Parameter Name	I/O ?	Value Type	Note
FEATURES	Input	ComplexValue or ComplexValueReference, GML2	A URL pointing to the data to be simplified GML from a WFS
TOLERANCE	Input	Literal xs:double	Units of measure is the unit of the CRS of the desired data  An example value used is 0.000062
SIMPLIFIED_FEATURES	Output	ComplexValue or ComplexValueReference, GML2	The resulting GML document produced by the generalization.

## 5.2.5 Simplification Results

OWS-4 demonstrated that the effect of data reduction using simplification on MSD3 road data was significant. The proposed simplification value of 0.000062 (measured in CRS-default units of EPSG:4326) results in a data reduction of approx. 40 percent of the original geometry data. The data reduction does not produce any cartographic errors, when the data has to be displayed as demonstrated in Figure 1.

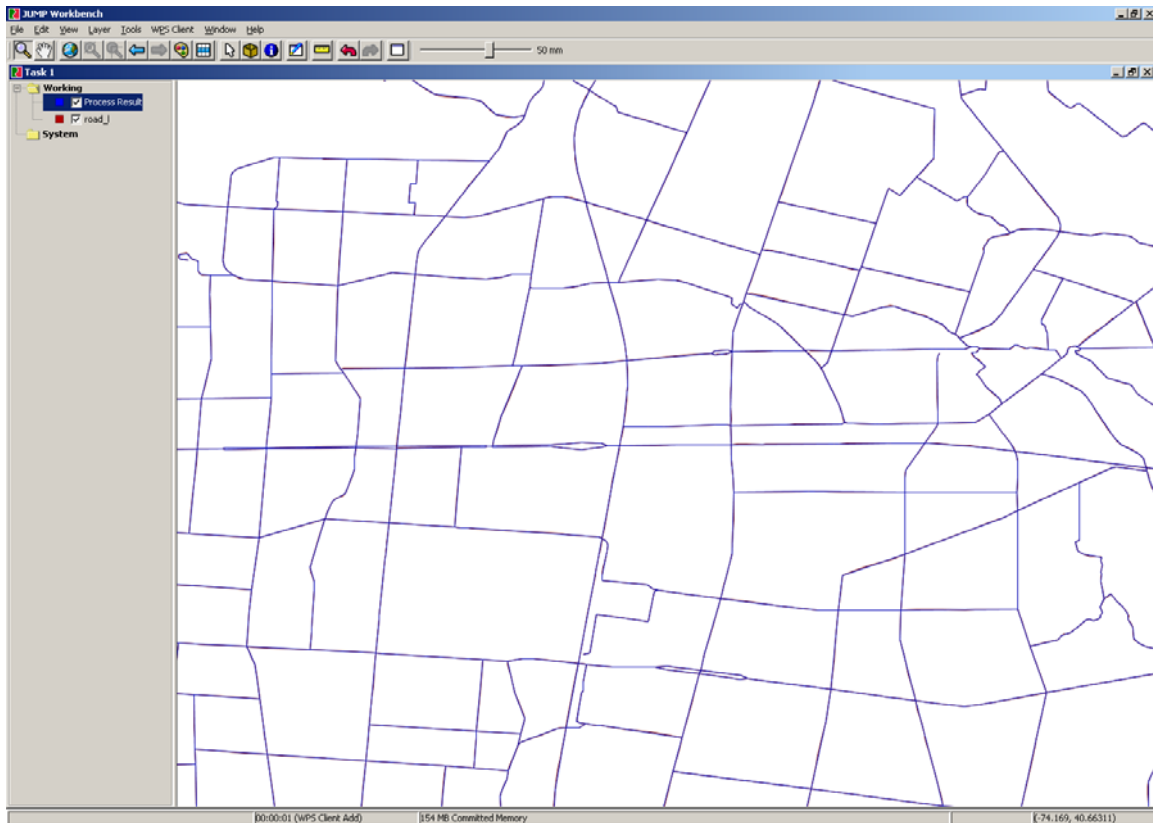


Figure 1: Overlay of the original and simplified MSD3 road data set with a data reduction of 40 %.

## 5.3 Clipping Process




### 5.3.1 Introduction

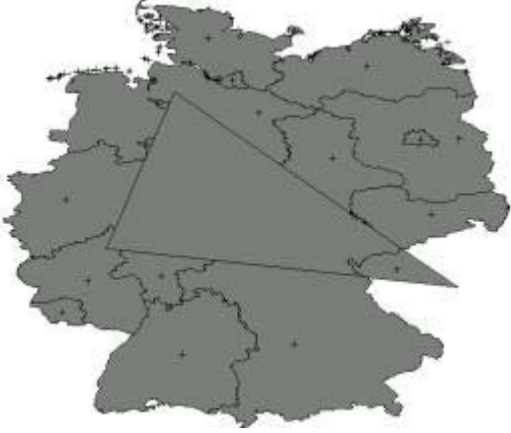
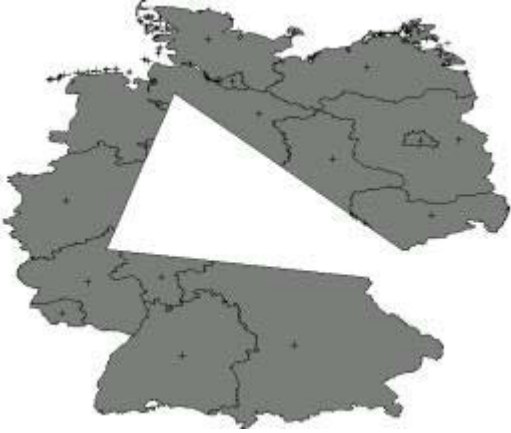
The purpose of the Web Clipping Service is to provide a means to clip GML feature data in different operations including OR, AND, NOT and XOR. GMU provides two kinds of WPS-Clipping. One is to read, clip and write GML feature data based on the Geographic Resources Analysis Support System ([GRASS](#)). The other one incorporates WFS-T to get, clip and update feature data based on the Java Topology Suite ([JTS](#)) and the WPS implementation from 52north. Table 1 is about some useful WPS-Clipping links. If the clipping operation fails for some reason, use WPS getErrors() functions to find the exact error message.

### 5.3.2 Operations

GRASS-based Clipping Service provides 4 clipping operations on GML 2.0, 3.1 and 3.1.1 as the following figures illustrate:

**Table 2: Clipping examples**

Clip region	
AND	
OR	

XOR	
NOT	

This version of the service implements the “fuzzy” clipping discussed, see [12] for a description of “fuzzy” clipping.

The inputs to the service are:

- **aInputURL**: original data address which can be a WFS get request. For example, [http://geobrain.laits.gmu.edu:8099/geoserver/wfs?SERVICE=WFS&REQUEST=GetFeature&TYPENAME=topp:statestable&Filter=<Filter xmlns='http://www.opengis.net/ogc'><BBOX><PropertyName>the\\_geom</PropertyName><gml:Box xmlns:gml='http://www.opengis.net/gml'><gml:coordinates>-117.294220,36.367267 -105.775974,43.654674</gml:coordinates></gml:Box></BBOX></Filter>](http://geobrain.laits.gmu.edu:8099/geoserver/wfs?SERVICE=WFS&REQUEST=GetFeature&TYPENAME=topp:statestable&Filter=<Filter xmlns='http://www.opengis.net/ogc'><BBOX><PropertyName>the_geom</PropertyName><gml:Box xmlns:gml='http://www.opengis.net/gml'><gml:coordinates>-117.294220,36.367267 -105.775974,43.654674</gml:coordinates></gml:Box></BBOX></Filter>)
- **bInputURL**: clipped area data address. For example, <http://geobrain.laits.gmu.edu:8099/temp/mdcutshp.gml>
- **ClipOperator**: operation name. For example, AND, OR, XOR and NOT.

### 5.3.3 WFS-T-based WPS-Clipping

This version of the service implements the “fuzzy” clipping as figure 1 illustrates. The algorithm used in brief is as following:

- 1) Check if the WFS has the featureType and the geometry field given as input.
- 2) Read the GML document containing the polygon.
- 3) Create another polygon by buffering it with the threshold given as the input.
- 4) Request WFS for all the feature which intersect the bounding box of the new polygon.
- 5) Out of all WFS features retrieved, filter out those features which do not fall completely inside(“covered by”) the new polygon.
- 6) Return the remaining features.

The inputs to the service are:

- End point of a WFS
- The feature type name corresponding to the features to be clipped
- The geometry field in the feature type
- A GML document containing a polygon corresponding to the clipping area. The polygon can be of any general shape.
- The fuzzy distance



“Fuzzy Clipping” is a type of feature selection that does not alter geometry

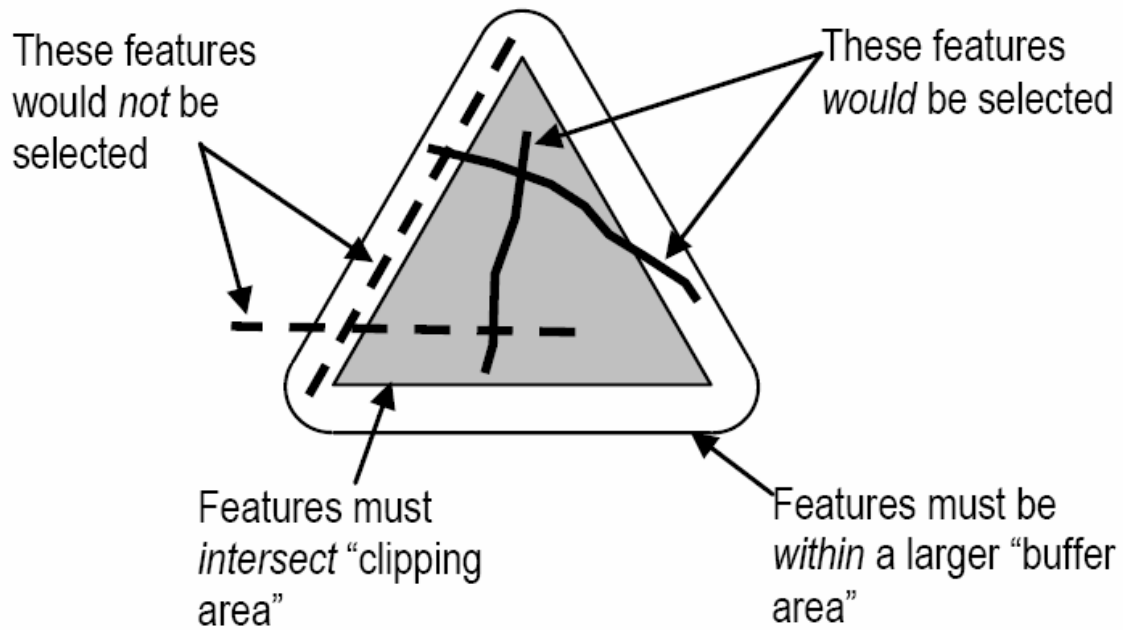


Figure 2: Fuzzy clipping

The following figures show the results of clipping a tiger data using a rectangular bounding box.

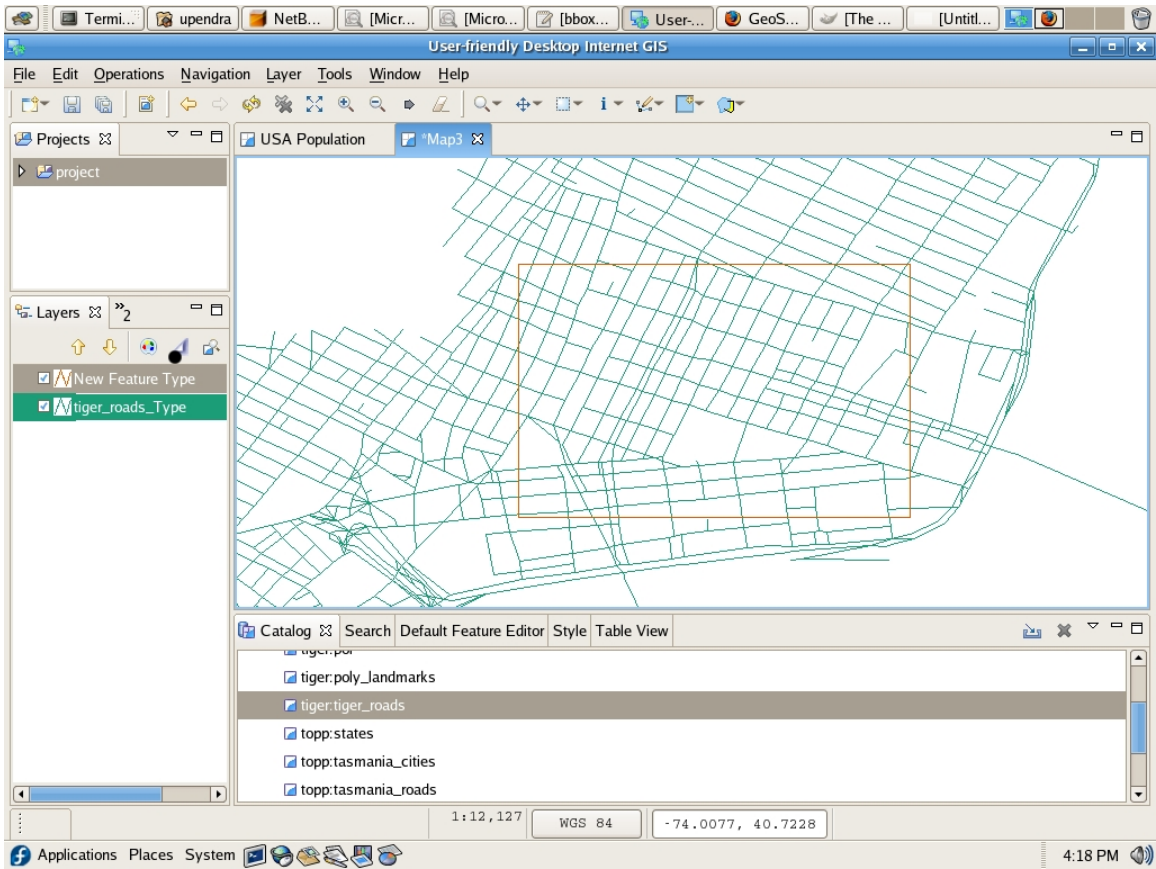
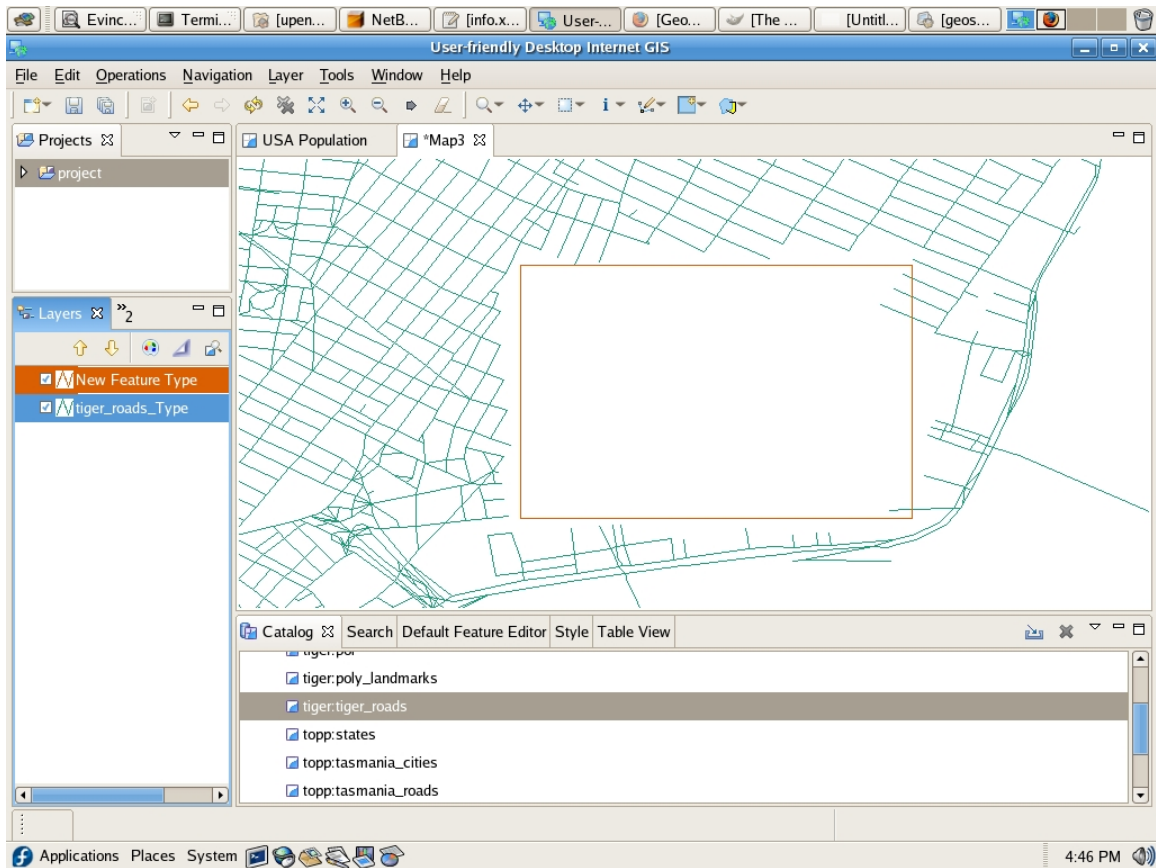


Figure 3: Before Clipping



**Figure 4: After clipping**

**Table 3: Clipping implementation links**

Get Capabilities	<a href="http://geobrain.laits.gmu.edu:8099/wps/WebProcessingService?Request=GetCapabilities&amp;Service=WPS">http://geobrain.laits.gmu.edu:8099/wps/WebProcessingService?Request=GetCapabilities&amp;Service=WPS</a>
Describe Process	<a href="http://geobrain.laits.gmu.edu:8099/wps/WebProcessingService?Request=DescribeProcess&amp;Service=WPS&amp;version=0.4.0&amp;identifier=clippingservice.clip">http://geobrain.laits.gmu.edu:8099/wps/WebProcessingService?Request=DescribeProcess&amp;Service=WPS&amp;version=0.4.0&amp;identifier=clippingservice.clip</a>
Test Page	<a href="http://geobrain.laits.gmu.edu:8099/wps/test_clip.html">http://geobrain.laits.gmu.edu:8099/wps/test_clip.html</a>
WSDL	<a href="http://geobrain.laits.gmu.edu:8099/axis/services/Clip_Functions?wsdl">http://geobrain.laits.gmu.edu:8099/axis/services/Clip_Functions?wsdl</a>

### 5.3.4 Service Demo using BPELPower:

Demo site: <http://geobrain.laits.gmu.edu:8099/bpel/admin/index.jsp>

Input examples:

- aInputURL: [http://geobrain.laits.gmu.edu:8099/temp/Maryland\\_Boundary.gml](http://geobrain.laits.gmu.edu:8099/temp/Maryland_Boundary.gml)
- bInputURL: <http://geobrain.laits.gmu.edu:8099/temp/mdcutshp.gml>

## 5.4 Binary Grid Process

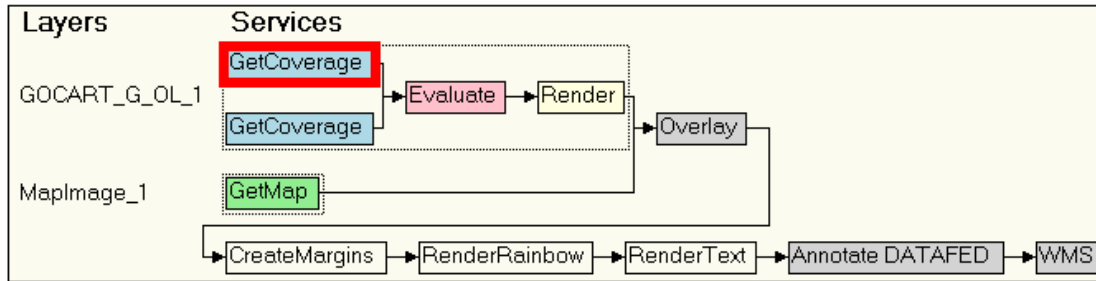
An objective of OWS-4 was to examine the use of WPS for processing output from a WCS and a SOS. The WCSes used for this test were the NASA GEOS Forecast Model and the GOCART Aerosol Model. The SOS used came from the NASA EO-1 Satellite hosted by Vightel. The WCS output was in the form of netCDF grids. The SOS served compressed GeoTiff images. Since outputs from both were in grid format, a binary grid processing service was created for testing the use of WPS in processing model output and satellite observations.

The grid inputs to the Binary Grid Processing Service can include WCS GetCoverage request or a SOS GetObservation request that returns an embedded URL to a grid file. The resultant grid from the processing service is accessible through a URL provided in the SOAP output. The output grid is specified as an input to the service. Other inputs to the binary processing service are described in Table 4.

**Table 4: Binary grid process parameters**

Parameters	I/O?	Value Type	Note
Grid A	Input	ComplexValueReference	WCS GetCoverage request (HTTP GET), SOS GetObservationrequest (HTTP GET), or other URL to grid file
Grid B	Input	ComplexValueReference	WCS GetCoverage request (HTTP GET), SOS GetObservationrequest (HTTP GET), or other URL to grid file
Selected Operation or user defined expression	Input	String	Available operations include: add, subtract, multiple, divide or user defined expressions such as $2*a - b$ .
Spatial bounding box in which operation should be executed	Input	BoundingBox	Options include: boundary of first grid, boundary of second grid, the union of the two grids, the intersection of the two grids, or a user defined bounding box
Spatial resolution of the output grid (x,y dim)	Input		Options include: the resolution of the first grid, the resolution of the second grid, or a user defined resolution.
Output Format	Input		Available as netCDF, GeoTiff, png, gif, jpeg, or DataFed binary formats
Result	Output		

The processing service was developed within the DataFed framework ([www.datafed.net](http://www.datafed.net)). DataFed is a web infrastructure based on OGC interoperability for fostering air quality and atmospheric science data access and analysis for air quality researchers, planners, and decision makers. The Figure below shows the binary processing service (indicated as the Evaluate box) within a DataFed service chain. Two GetCoverage requests serve as the grid inputs. The processing service settings are defined through form input. The output is passed through various rendering, annotation, and overlay services.



**Figure 5: DataFed service chain**

The binary processing service was used in the OWS-4 Earth Observation Demo as described in Annex F.

## 5.5 Feature Fusion Process

The goal of this project is to make a GML Feature Fusion Service using the WPS. The GML Feature Fusion Service was formerly known as the Data Aggregation Service (DAS) developed during OWS-3. The DAS was originally developed as an extension to a WFS. The [OWS-3 Data Aggregation Service Interfaces](#) document discusses the interface.

For OWS-3 the DAS was implemented as a WFS. That WFS had a new operation used to set the mapping file.

The GML Feature Fusion Service's (FFS) purpose is to fuse one or more heterogeneous GML feature collections into one homogeneous feature collection. When there is only one input feature collection this service simply performs a GML transformation.

This document will use the acronym FFS to refer to the service implemented within a WPS. The DAS acronym will be used to refer to the WFS implementation of the service.

There are two ways of aggregating GML features:

- Aggregation of two distinct data sets, describing different spatial objects
- Aggregation of two data sets, which describe same spatial objects, but different aspects of these objects. The sameness of the objects is identified via object identifiers (or other properties or policies). This is similar to relational database joins (see Subclause 5.5.4).

### 5.5.1 Requirements

The service requires the following list of inputs and outputs.

#### Inputs

- List of GML feature collections to fuse. The list is actually a list of URL to GML instance documents. Those instance documents can come from a WFS or any other location available on the internet.
- Mapping Definition File - used to map the feature collections to an existing feature collection.
- Target Schema - the schema of the resulting GML instance document? This input is not likely to be necessary. It is just added for discussion purposes.

## **Outputs**

- A GML feature collection
- The application schema of the resulting feature collection? This output might not be necessary.

### **5.5.2 Schema Mapping and Assembly Tool (SMAT)**

The FFS requires a mapping file created using the schema mapping and assembly tool. Currently the SMAT only maps to existing schemas. It does not generate a new schema. Would the capability to generate a new schema be worthwhile?

### **5.5.3 Feature Collection Transformation and Feature Collection Appending Services**

The FFS can be broken down into two smaller services:

- A Feature Collection Transformation Service (FCTS) that has only one source feature collection and schema and results in one output feature collection in the target schema.
- A Feature Collection Appending Service (FCAS) that merges many homogeneous (same schema) feature collections into one large feature collection valid against the same schema as the source collections.

The FFS could then use the two services (FCTS and FCAS) to do its work. However, it would require a change to the FFS interface.

There are two ways to use the FCTS and FCAS:

1. Each input feature collection is first transformed using the FCTS then the data is merged using a FCAS. This is the simplest and it seems that the existing DAS uses this technique.
2. The input feature collections are joined to a 3<sup>rd</sup> private structure schema and then transformed to the target schema. In more general, the data sets are joined using a certain policy. A general policy could also describe a conflation process (objects have the same geometry, but not other common properties). After joining both

data sets to a single one, the data set could be transformed according to the target data model.

#### 5.5.4 XML Joins and Unions

The concepts described by relational databases such as join and union can be extended to XML and GML. The analogous concepts are:

- The database schema is similar to a GML application schema
- A feature collection is the set of records in a database
- Filters in the WFS deal with sub setting the features and the application schema. Selection in SQL also deals with sub setting both the records and the schema.
- In SQL the join concept exists between tables but the OGC hasn't discussed it yet in GML. A search on the web finds several documents discussing XML joins.

When joining documents the following parameters need to be defined:

- The input documents - equivalent to the list of tables in an SQL select statement
- The output schema - similar to the "as" statement in an SQL select statement. The difference here is that SQL uses the relational model which is well defined and allows one to make certain assumptions in the language. With XML, the output schema is less constrained in its structure resulting in a very complex mapping.
  - Subset of only one of the input schemas
  - Some merging of the input schemas
  - Some completely arbitrary schema
- A filtering predicate - the "where" clause. It is used to choose the features that are to be put into the output document. It does so by
  - placing conditions on feature properties using various types of operators, for example: featureA/propertyB > 10
  - determining the join condition, for example: featureA/id == featureB/id

#### 5.5.5 Data models and "governance"

The use-case for feature-fusion is interesting.

Who is responsible for governance of the various schemas? In particular, are the schemas for both the sources and target WFS local or "private"? If so then it is the client's

responsibility (who presumably has specified the target schema) to manage mappings to the schema.

OTOH, if the target schema is a standard or "community" schema, then deploying a Feature Fusion Service to convert various sources is an example of a "mediator" architecture.

However, it is arguable that, if possible, the job of transforming the schema from a private/local model to a community model should be pushed back to the data custodian. After all they know their data and schema best, and are probably in the best position to manage the definition of a transformation mapping to a community model - that is, if they are interested in publishing a standard data product. In that case, the FFS could be deployed as a "wrapper" service at the data custodian's end. The data custodian is not always willing to do the transformation. In such a case a third party has to develop the transformation

There are various architectures and implementations for generating a "standard" data product. A FFS is a nice piece to have, but its deployment position should be considered carefully.

### **5.5.6 Questions**

1. How do we encode the list of input feature collections? The WPS doesn't directly support a list of URL's so a new type in the WPS defining a way to accept a "list" may be required. Another option is to "hack" it as a simple XML file or a comma delimited list of URL's?
2. What actually is the use of such a service, if the mapping file is an input parameter? Then the client is actually aware of the mapping and could just do it on its own. That would likely provide better performance and perhaps more reliable. I would perhaps prefer something, which is actually more open. Something like indicated in the WCTS. Actually as I now think about transformation and mapping, this leads directly to the idea of mobile code. A mapping is submitted to the service as a parameter. This mapping could be a XSLT script or a real java class. We should really think about, the persistence of such scripts, a client could indicate if such a transformation-script is permanent stored or not. This would be more open and more flexible. And now back to processes, is not a geo-process a transformation of spatial data? So this idea should be reflected in the WPS specification.

## **6 WPS, WSDL, and SOAP**

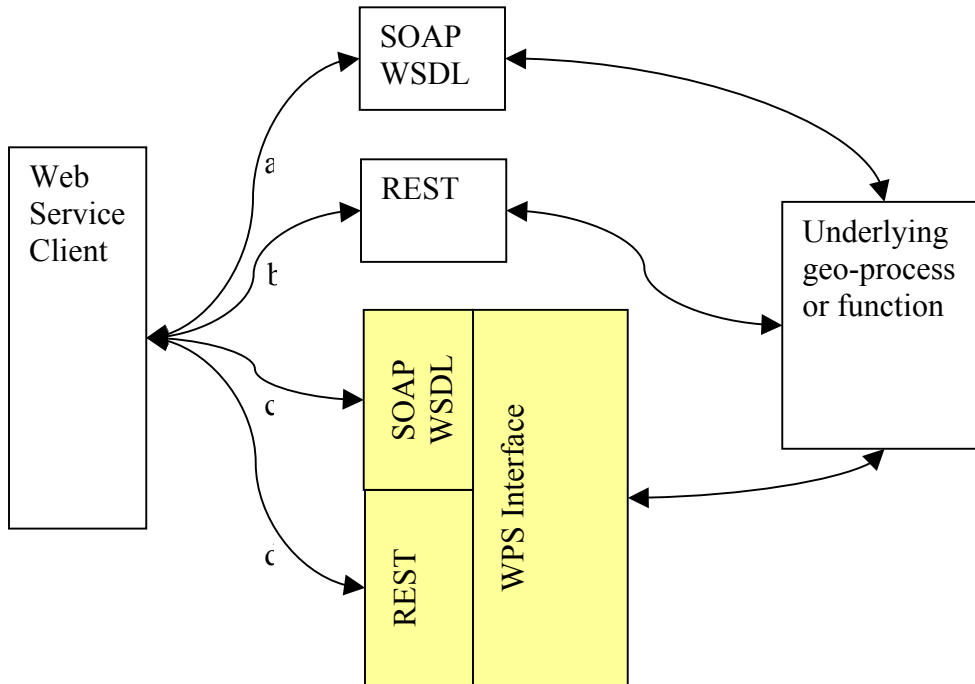
### **6.1 Introduction**

One of the OWS-4 project's mandates is to study and improve the WSDL usage in OGC services. Since the WPS and WSDL have overlapping responsibilities; and since the



WPS is different than most OGC services, more effort was required to devise a reasonable WSDL strategy for the WPS.

## 6.2 Web enabling a geo-process



**Figure 6: Web enabling a geo-process**

Figure 6 shows the four ways that the OGC has envisaged web enabling geo-processes.

- a) Use a custom SOAP/WSDL architecture
- b) Use a custom REST architecture
- c) Use a standard WPS interface along with a SOAP WSDL architecture
- d) Use a standard WPS interface along with a REST architecture
- e) A fifth scenario exists. It is described in subclause 6.2.1

The first two architectures are only interoperable at the transport protocol level and possibly at higher levels. However, they are not interoperable at the application protocol level. The last two are interoperable at all levels up to the application protocol level.

### 6.2.1 SOAP & WSDL in the Binary Grid Processing Service <sup>1</sup>

The WPS can be interpreted as an extension of WSDL. It defines the geospatial aspects of a processing service that is described and bound using WSD and SOAP. From this perspective, WPS could be viewed as consisting of two components:

1. Set of guidelines for SOAP/WSDL
2. REST interface to SOAP/WSDL

The guidelines would provide best practices and standardized implementations of WSDL. WSDL offers a wide range of possible implementations to the developer. A standardized implementation of WSDL for geo-processing services (defined as the WPS) would help interoperability among geo-processing services.

In this case, the procedure for creating a geo-processing service is:

1. create the processing service using a WSDL description (according to the WPS WSDL schemas)
2. provide REST interfaces for GetCapabilities, DescribeProcess, and Execute

Users of the geo-processing service will have options to interface with the service directly through SOAP/WSDL or through a REST interface on top of the SOAP/WSDL (see figure below).

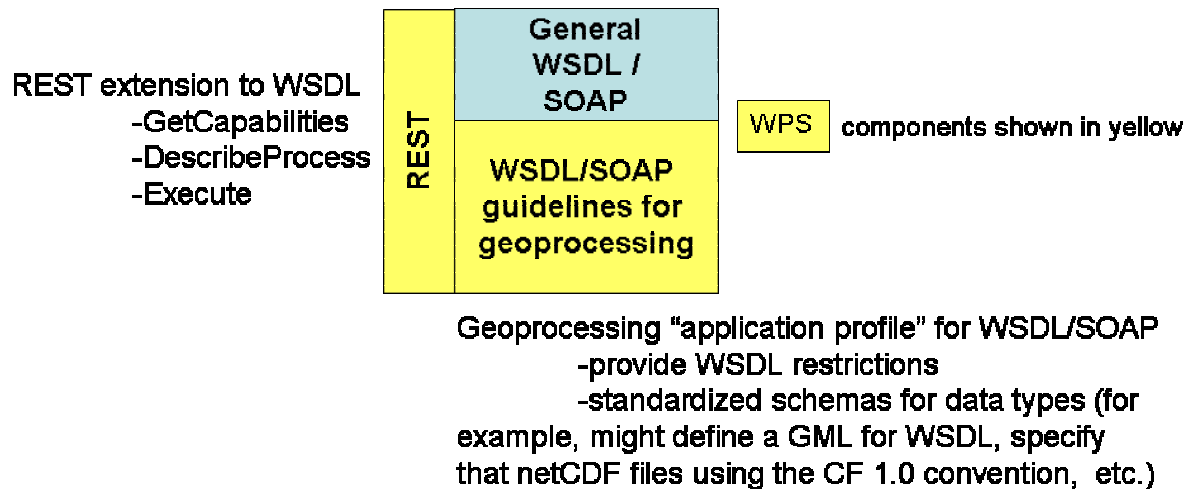


Figure 7: REST, WSDL/SOAP, and the WPS

<sup>1</sup> This subclause is copied directly from [11].

### **6.3 WPS WSDL document style**

#### **6.3.1 WPS publishes one all-encompassing WSDL document**

The idea with this approach is that the WPS publishes only one WSDL document. That WSDL document describes the WPS operations available like GetCapabilities, DescribeProcess, and Execute. Additionally, the WSDL document describes each process available.

GMU has kindly assembled a sample WSDL document that fits into this design. See Annex C

#### **6.3.2 WPS publishes several WSDL documents**

Another approach is that the WPS publishes several WSDL documents, one for each process available within the WPS and an additional one describing the WPS service. The WPS service WSDL can include the individual WSDL documents. The benefit of this is that the WSDL documents are uncoupled, smaller, and easier to understand. Also, a client not knowing the WPS interface should be able to execute a WPS process.

#### **6.3.3 WSDL and the OGC Capabilities**

Related to the question of using a fully formed URL or a base URL is the fact that clients should be able to choose their favourite technique for retrieving the service description. Some clients prefer WSDL while others prefer OGC Capabilities documents. Thus, for clients that prefer WSDL, it is important they be able to determine the WSDL location URL from a capabilities URL or the capabilities document. And vice versa, for clients that prefer a capabilities document, it is important they be able to determine the capabilities location from the WSDL

There are two solutions to this:

1. Provide a base URL with which both the WSDL location URL and the Capabilities can be derived.
2. Refer to the WSDL within the capabilities and vice versa, refer to the capabilities within the WSDL.

Both of these solutions suffer from the fact that a WSDL document can describe more than one service while a capabilities document can only describe one service.

### **6.4 WSDL location URL**

#### **6.4.1 Overview**

A WSDL aware WPS client needs a WSDL location URL to access the processes available on a WPS server. How is that WSDL location URL discovered? There are

many mechanisms for discovering a WSDL location URL: via a service registry, sent via email, using a web search tool like Google, or some other mechanism. However, all of these mechanisms can result in:

1. A base URL which needs to be augmented using a standard construction mechanism to create new URL. The new URL is used to retrieve the WSDL document.
2. A fully formed WSDL location URL that can be used as is to retrieve the WSDL document.

How does a WSDL aware WPS client determine the URL to access a WSDL document describing an OGC service?

#### 6.4.2 WSDL location URL construction mechanisms

Another way of putting this question is “how is the WSDL location URL constructed from a base URL such that the WSDL can be consistently retrieved?” The OWS-4 GPW thread discussed this and the conclusion was to wait until WSDL 2 is released. However, the current draft version of the WSDL 2 specification does not state how to retrieve a WSDL document. Likely, it will never mandate a technique for determining the WSDL document location. The reason is that there are too many construction techniques and mandating such a rule could break existing implementations.

Having said that, the OGC uses the URL construction for KVP requests to OWS instances, and thus it is reasonable to expect the same treatment for WSDL documents describing OGC services.

- Add a request=GetWsd1 request to OGC specifications (this option is described in 05-008r1 OWS Common). For example:
  - `http://www.yourserver.com/path?request=GetWsd1`
- Provide a standard URL syntax for accessing the WSDL relative to the OGC service. The following have been mentioned:
  - `http://www.yourserver.com/path/`
  - `http://www.yourserver.com/path?wsdl`
  - `http://www.yourserver.com/path/path.wsdl`
  - `http://link.to.the.service/wsdl#`
  - <http://link.to.the.service/wsdl?>

### 6.4.3 Fully formed WSDL location URL

For the most part, the question of how to construct a WSDL location URL is irrelevant because the very idea that the WSDL location URL must be constructed from a base URL is flawed.

A better approach is that the OGC provide a best practices guideline suggesting a technique for formulating full WSDL location URLs. Operators can then register their WPS servers, or any OGC service, with a registry using the full WSDL location URL.

One benefit of using a fully formed WSDL location URL is that there is no need for client implementers to write special code to construct the final URL from the base URL. Also, there is no question about what constitutes a base URL.

### 6.5 Requires further investigation

The following subjects were discussed briefly related to creating WSDL descriptions for the WPS. It is important that further discussions be started to resolve them.

- f) The WPS DescribeProcess operation is similar to a WSDL document. Using an XSLT, it may be possible to transform from one to the other and vice versa. Likely, the transformation is not one-to-one and thus there will be a loss of data in one or both directions.
- g) The WPS Execute operation is the entree point into a WPS process - it's how the process is actually invoked. When creating a WSDL for a particular process that is the main operation that needs to be described. It is possible that that is the only operation that needs to be described within the WSDL for a WPS process.

### 6.6 SOAP

Using SOAP within the WPS was not discussed at length by the GPW thread in OWS-4. However, with the few discussions we had, we decided that the WPS should use the SOAP mechanism(s) already espoused by the OGC from previous OWS projects. Having said that, there is no clear consensus yet and this discussion needs to remain open.

## Annex A Workflows

### A.1 GPW Workflow Summary

The GPW thread developed the following workflows:

- Data reduction workflow - reduces the quantity of data transferred to a client
- Feature Update Workflow - provides feature updates and topological quality assessment

Both workflows use a BPEL engine with a WFS or WFS-T adapter.

#### A.1.1 Data reduction workflow

The clip-zip-ship scenario was devised to reduce the quantity of feature data sent to a light weight client in the field. To further reduce the data we added the possibility of doing generalization in the workflow.

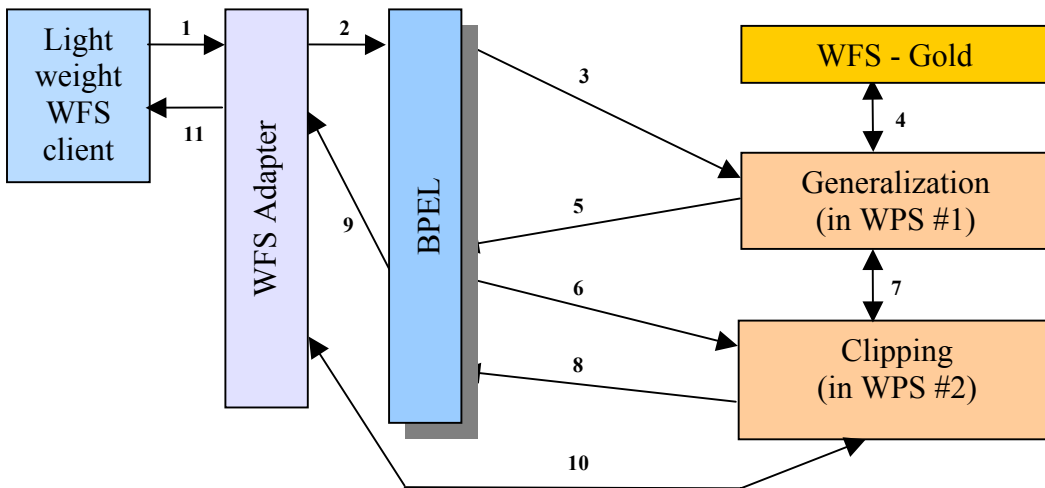


Figure 8: Data reduction workflow sequence

Figure 8 shows the sequence that services are activated within the data reduction workflow.

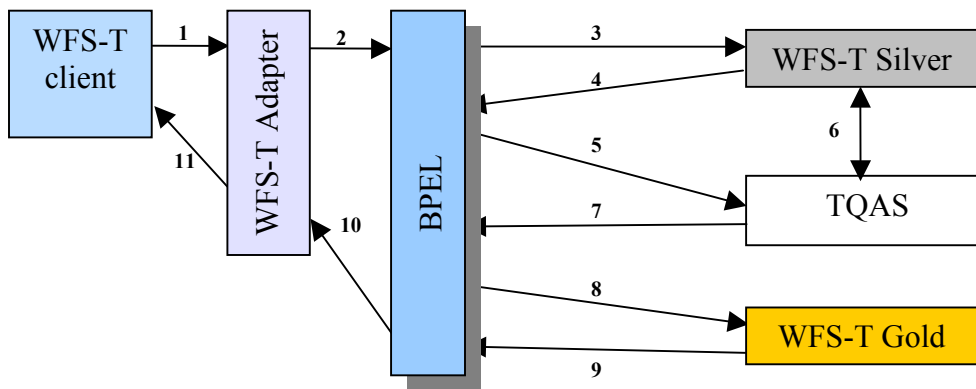
Table 5: Data reduction workflow

Step	Description	Interface
1	Client requests features from WFS adapter, with or without filter	WFS GetFeature request
2	Starts the data reduction BPEL script.	??? SOAP?
3	Script calls Generalization process with reference to Gold WFS as input parameter.	WPS Execute request Ideally with store=true

4	Request to Gold WFS to retrieve features to generalize. The result is a GML document matching the filter sent in step 1.	WFS GetFeature request and response pair.
5	Response from generalization process. This should only contain references to the results. In other words the results are stored on the WPS server that performed the generalization. This reference can then be forwarded to the clipping in the next step.	WPS Execute response
6	BPEL script sends request to clipping process. It passes the reference to the results from the generalization process.	WPS Execute request Ideally with store=true
7	Request and response to WPS server #1 that performed generalization to retrieve generalization results.	HTTP GET using URL reference returned in step 6.
8	Response from clipping service. This should contain a reference to the results stored on the WPS #2.	WPS Execute request Ideally with store=true
9	Response from BPEL script. BPEL script tells WFS Adapter where to get the clipping results. The clipping results should be stored on the WPS #2.	SOAP
10	WFS Adapter retrieves clipped GML document from clipping service.	HTTP GET using URL reference returned in step 8.
11	GML document containing clipped results are sent to WFS client	WFS Response, GML document

**A.1.2 Synchronous feature update workflow**

The feature update workflow ingests WFS transactions. A WFS transaction passes the features through a topological quality assessment service where they are evaluated.



**Figure 9: Synchronous feature update workflow**

Figure 10 shows a synchronous workflow.

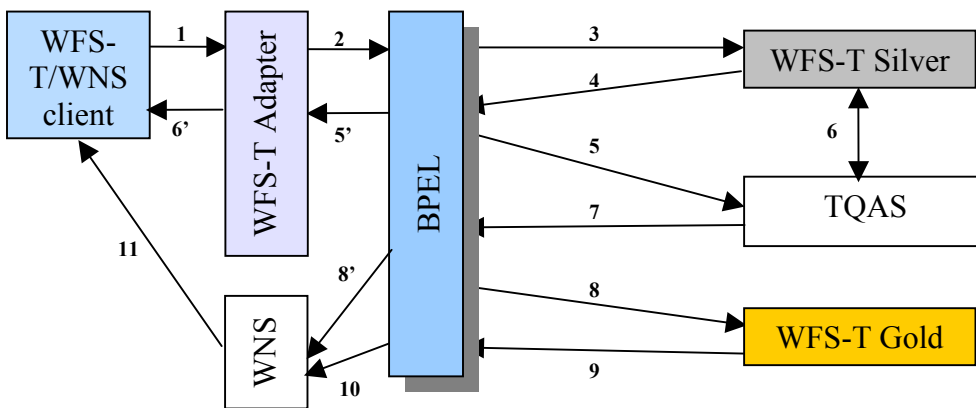
**Table 6: Synchronous feature update workflow**

Step	Description	Interface
1	Client requests transaction to WFS adapter.	WFS-T Transaction request
2	Starts the feature update BPEL script.	SOAP
3	BPEL forwards transaction to WFS-T Silver	WFS-T Transaction request
4	WFS-T transaction response	

5	Assuming transaction succeeded, TQAS service is hit	SOAP
6	TQAS interacts with WFS-T Silver to perform quality assessment	WFS GetFeature
7	Response from TQAS	SOAP and ISO 19139
8	Apply transaction to WFS-T Gold	
9	Whether transaction on WFS-T Gold succeeded or not	
10	Respond to WFS-T Adapter	SOAP
11	Notify client WFS-T client of result	

**A.1.3 Asynchronous feature update workflow**

The feature update workflow ingests WFS transactions. A WFS transaction passes the features through a topological quality assessment service where they are evaluated.



**Figure 10: Feature update workflow**

Figure 10 shows the sequence of an asynchronous transaction implemented as a workflow. Note that Figure 10 does not use a WPS; it is only here for informational purposes.

**Table 7: Asynchronous feature update workflow**

Step	Description	Interface
1	Client requests transaction to WFS adapter.	WFS Transaction request
2	Starts the feature update BPEL script.	SOAP
3	BPEL forwards transaction to WFS-T Silver	WFS-T Transaction request
4	WFS-T transaction response	
5	Assuming transaction succeeded, TQAS service is hit	SOAP
5'	WFS-T transaction response – whether succeeded or failed on silver WFS-T	WFS transaction silver
6	TQAS interacts with WFS-T Silver to perform quality assessment	WFS GetFeature
6'	A response signalling that the transaction was accepted	Currently impossible with WFS interface



- 7 Response from TQAS SOAP and ISO 19139
- 8 Apply transaction to WFS-T Gold
- 8' When TQAS fails notify WNS to signal client
- 9 Whether transaction on WFS-T Gold succeeded or not
- 10 Send status from step #9 to WNS
- 11 Notify client

#### **A.1.4 Issues**

WFS-T doesn't support long/asynchronous transactions via a WNS. It is currently impossible, with the WFS-T interface, to send a request to the WFS and be informed at a later date (asynchronously) whether the status succeeded or not.

Due to the nature of HTTP and all of the firewalls it is difficult to use and implement asynchronous mechanisms in web services.

## **Annex B Change Requests**

### **B.1 Change request - Support POST in ComplexValueReference element**

This draft change request, titled “[06-151- WPS Change Request - Support POST in ComplexValueReference](#)” is available on the OGC Portal at:  
[http://portal.opengeospatial.org/files/?artifact\\_id=18557&version=1](http://portal.opengeospatial.org/files/?artifact_id=18557&version=1)

## Annex C WSDL Examples

### C.1 GMU WPS WSDL

The following XML is taken directly from the GMU WPS running the clipping service required for the data reduction workflow. This version was copied from:

<http://geobrain.laits.gmu.edu/mpgc/wps/0.4.0/wps.wsdl>.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:wps="http://www.opengeospatial.net/wps"
xmlns:tns="http://geobrain.laits.gmu.edu/schemas/wpsWSDL"
targetNamespace="http://geobrain.laits.gmu.edu/schemas/wpsWSDL">
  <types>
    <xsd:schema>
      <xsd:import namespace="http://www.opengeospatial.net/wps"
schemaLocation="http://geobrain.laits.gmu.edu/mpgc/wps/0.4.0/wpsCommon.xsd"/>
      <xsd:import namespace="http://www.opengeospatial.net/wps"
schemaLocation="http://geobrain.laits.gmu.edu/mpgc/wps/0.4.0/wpsGetCapabilities.xsd"/>
      <xsd:import namespace="http://www.opengeospatial.net/wps"
schemaLocation="http://geobrain.laits.gmu.edu/mpgc/wps/0.4.0/wpsDescribeProcess.xsd"/>
      <xsd:import namespace="http://www.opengeospatial.net/wps"
schemaLocation="http://geobrain.laits.gmu.edu/mpgc/wps/0.4.0/wpsExecute.xsd"/>
    </xsd:schema>
  </types>
  <message name="GetCapaMessageIn">
    <part name="request" type="wps:RequestType"/>
    <part name="service" type="wps:ServiceType"/>
    <part name="AcceptVersions" type="wps:VersionType"/>
    <part name="Sections" type="wps:SectionType"/>
    <part name="updatesequence" type="xsd:string"/>
    <part name="AcceptFormats" type="xsd:string"/>
  </message>
  <message name="GetCapaMessageOut">
    <part name="parameters" element="wps:Capabilities"/>
  </message>
  <message name="DescribeProcessMessageIn">
    <part name="request" type="wps:RequestType"/>
    <part name="service" type="wps:ServiceType"/>
    <part name="version" type="wps:VersionType"/>
    <part name="Identifier" type="xsd:string"/>
  </message>
  <message name="DescribeProcessMessageOut">
    <part name="parameters" element="wps:ProcessDescriptions"/>
  </message>
  <message name="ExecuteMessageIn">
    <part name="parameters" element="wps:Execute"/>
  </message>
  <message name="ExecuteMessageOut">
    <part name="parameters" element="wps:ExecuteResponse"/>
  </message>
  <message name="WPSException">
    <part name="exception" type="xsd:string"/>
  </message>
  <portType name="WPS_HTTP_Get_Port">
    <operation name="GetCapabilities">
      <input message="tns:GetCapaMessageIn"/>
      <output message="tns:GetCapaMessageOut"/>
      <fault name="exception" message="tns:WPSException"/>
    </operation>
  </portType>

```

```

</operation>
<operation name="DescribeProcess">
  <input message="tns:DescribeProcessMessageIn"/>
  <output message="tns:DescribeProcessMessageOut"/>
  <fault name="exception" message="tns:WPSEException"/>
</operation>
</portType>
</portType>
<portType name="WPS_HTTP_Post_Port">
  <operation name="Execute">
    <input message="tns:ExecuteMessageIn"/>
    <output message="tns:ExecuteMessageOut"/>
    <fault name="exception" message="tns:WPSEException"/>
  </operation>
</portType>
<binding name="WPS_HTTP_GET_Binding" type="tns:WPS_HTTP_Get_Port">
  <http:binding verb="GET"/>
  <operation name="GetCapabilities">
    <http:operation location=""/>
    <input>
      <http:urlEncoded/>
    </input>
    <output>
      <mime:content part="response" type="text/xml"/>
    </output>
    <fault name="exception">
      <mime:content type="text/xml"/>
    </fault>
  </operation>
  <operation name="DescribeProcess">
    <http:operation location=""/>
    <input>
      <http:urlEncoded/>
    </input>
    <output>
      <mime:content type="text/xml"/>
    </output>
    <fault name="exception">
      <mime:content type="text/xml"/>
    </fault>
  </operation>
</binding>
<binding name="WPS_HTTP_POST_Binding" type="tns:WPS_HTTP_Post_Port">
  <http:binding verb="POST"/>
  <operation name="Execute">
    <http:operation location=""/>
    <input>
      <!--<mime:content type="application/x-www-form-urlencoded"/>-->
      <mime:content type="text/xml"/>
    </input>
    <output>
      <mime:content part="response" type="text/xml"/>
    </output>
    <fault name="exception">
      <mime:content type="text/xml"/>
    </fault>
  </operation>
</binding>
<service name="WPS_Generalization">
  <port name="WPS_Generalization_GET" binding="tns:WPS_HTTP_GET_Binding">
    <http:address location="http://geoserver.itc.nl:8080/wps/WebProcessingService"/>
  </port>
  <port name="WPS_Generalization_POST" binding="tns:WPS_HTTP_POST_Binding">
    <!--<http:address location="http://geoserver.itc.nl:8080/wps/WebProcessingService"/>-->
    <http:address location="http://data.laits.gmu.edu:8440/wps/WebProcessingService"/>
  </port>
</service>
<!-- *****
                        Messages
***** -->
<!-- *****
                        Ports

```

```

***** -->
<!-- *****
          Bindings
***** -->
<!-- *****
          Services
***** -->
</definitions>

```

## C.2 Binary Grid Processing Service WSDL

```

<?xml version="1.0" encoding="utf-8"?>
<definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:bgo="http://datafed.net/xs/MapGridOperator"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  targetNamespace="http://datafed.net/xs/MapGridOperator"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <documentation>Performs operations on grids.</documentation>
  <types>
    <xs:schema elementFormDefault="qualified"
      xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:cube="http://datafed.net/xs/Cube"
      targetNamespace="http://datafed.net/xs/MapGridOperator"
      xmlns:dp="http://datafed.net/xs/DataPrimitives"
      xmlns="http://datafed.net/xs/MapGridOperator">
      <xs:import namespace="http://datafed.net/xs/Cube" schemaLocation="Cube.xsd" />
      <xs:import namespace="http://datafed.net/xs/DataPrimitives"
        schemaLocation="DataPrimitives.xsd" />
      <xs:element name="Add">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="grid0" type="cube:CubeType" />
            <xs:element name="grid1" type="cube:CubeType" />
            <xs:element name="GridAdjustSettings"
              type="GridAdjustSettingsType" />
            <xs:element name="format" type="cube:CubeFormatType"
              default="internal" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="Subtract">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="grid0" type="cube:CubeType" />
            <xs:element name="grid1" type="cube:CubeType" />
            <xs:element name="GridAdjustSettings"
              type="GridAdjustSettingsType" />
            <xs:element name="format" type="cube:CubeFormatType"
              default="internal" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="Multiply">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="grid0" type="cube:CubeType" />
            <xs:element name="grid1" type="cube:CubeType" />
            <xs:element name="GridAdjustSettings"
              type="GridAdjustSettingsType" />
            <xs:element name="format" type="cube:CubeFormatType"
              default="internal" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="Divide">
        <xs:complexType>
          <xs:sequence>

```

```

        <xs:element name="grid0" type="cube:CubeType" />
        <xs:element name="grid1" type="cube:CubeType" />
        <xs:element name="GridAdjustSettings"
type="GridAdjustSettingsType" />
        <xs:element name="format" type="cube:CubeFormatType"
default="internal" />
    </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Evaluate">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="expression" type="xs:string" />
            <xs:element name="grid0" type="cube:CubeType" />
            <xs:element name="grid1" type="cube:CubeType" />
            <xs:element name="GridAdjustSettings"
type="GridAdjustSettingsType" />
            <xs:element name="format" type="cube:CubeFormatType"
default="internal" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="ZeroGridToNaNGrid">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="grid" type="cube:CubeType" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="MakeVector2D">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="output_param_abbr" type="xs:string" />
            <xs:element name="units" type="xs:string" />
            <xs:element name="style" type="VectorStyles" />
            <xs:element name="grid0" type="cube:CubeType" />
            <xs:element name="grid1" type="cube:CubeType" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:simpleType name="VectorStyles">
    <xs:restriction base="xs:string">
        <xs:enumeration value="U_V" />
        <xs:enumeration value="mag_dir" />
        <xs:enumeration value="mag" />
        <xs:enumeration value="dir" />
    </xs:restriction>
</xs:simpleType>
<xs:complexType name="GridAdjustSettingsType">
    <xs:sequence>
        <xs:element name="Dimensions" type="xs:string" default="lat lon" />
        <xs:element name="ZoomStyle" type="GridZoomAdjustStylesType"
default="must_match" />
        <xs:element name="Zoom" type="dp:mlstring" default="" />
        <xs:element name="SizeStyle" type="GridSizeAdjustStylesType"
default="use_default" />
        <xs:element name="Size" type="dp:mlstring" default="" />
    </xs:sequence>
</xs:complexType>
<xs:simpleType name="GridZoomAdjustStylesType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="must_match" />
        <xs:enumeration value="use_first" />
        <xs:enumeration value="use_second" />
        <xs:enumeration value="union" />
        <xs:enumeration value="intersection" />
        <xs:enumeration value="user_defined" />
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="GridSizeAdjustStylesType">
    <xs:restriction base="xs:string">

```

```

        <xs:enumeration value="use_default" />
        <xs:enumeration value="use_finer" />
        <xs:enumeration value="use_coarser" />
        <xs:enumeration value="user_defined" />
    </xs:restriction>
</xs:simpleType>
<xs:element name="Response">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="cube:Cube" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
<xs:schema elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:lin="http://datafed.net/xs/Lineage"
targetNamespace="http://datafed.net/xs/Cube" xmlns="http://datafed.net/xs/Cube">
    <xs:import namespace="http://datafed.net/xs/Lineage"
schemaLocation="Lineage.xsd" />
    <xs:element name="Cube" type="CubeType" />
    <xs:complexType name="CubeType">
        <xs:sequence>
            <xs:element name="CubeRef" type="CubeRefType"
default="http://dataserver.net/uri_pointing_to_NetCDF_with_CF2_conventions" />
            <xs:element ref="lin:Statistics" minOccurs="0" />
            <xs:element ref="lin:LineageList" minOccurs="0" />
        </xs:sequence>
    </xs:complexType>
    <xs:simpleType name="CubeFormatType">
        <xs:annotation>
            <xs:appinfo>xs_handler:CAPITA.Wui,CAPITA.Wui.CubeFormats</xs:appinfo>
        </xs:annotation>
        <xs:restriction base="xs:string" />
    </xs:simpleType>
    <xs:simpleType name="CubeRefType">
        <xs:restriction base="xs:anyURI" />
    </xs:simpleType>
</xs:schema>
<xs:schema elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://datafed.net/xs/DataPrimitives"
xmlns="http://datafed.net/xs/DataPrimitives">
    <xs:simpleType name="CSVTextType">
        <xs:restriction base="xs:string" />
    </xs:simpleType>
    <xs:simpleType name="DatasetAbbrType">
        <xs:restriction base="xs:string" />
    </xs:simpleType>
    <xs:simpleType name="DateTimeType">
        <xs:union memberTypes="xs:dateTime xs:date" />
    </xs:simpleType>
    <xs:simpleType name="DateTimeListType">
        <xs:restriction base="xs:string" />
    </xs:simpleType>
    <xs:simpleType name="DateTimeNowType">
        <xs:union memberTypes="xs:string xs:dateTime xs:date" />
    </xs:simpleType>
    <xs:complexType name="DateTimeRangeType">
        <xs:sequence>
            <xs:element name="time_min" type="DateTimeType" default="1970-01-
01T00:00:00" />
            <xs:element name="time_max" type="DateTimeNowType" default="now" />
        </xs:sequence>
    </xs:complexType>
    <xs:simpleType name="TimeUnitType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="year" />
            <xs:enumeration value="month" />
            <xs:enumeration value="week" />
            <xs:enumeration value="day" />
        </xs:restriction>
    </xs:simpleType>

```

```

        <xs:enumeration value="hour" />
        <xs:enumeration value="minute" />
        <xs:enumeration value="second" />
    </xs:restriction>
</xs:simpleType>
<xs:complexType name="LatLonType">
    <xs:sequence>
        <xs:element name="lat" type="xs:float" default="40" />
        <xs:element name="lon" type="xs:float" default="-90" />
    </xs:sequence>
</xs:complexType>
<xs:complexType name="LatLonRangeType">
    <xs:sequence>
        <xs:element name="lat_min" type="xs:float" default="-90" />
        <xs:element name="lat_max" type="xs:float" default="90" />
        <xs:element name="lon_min" type="xs:float" default="-180" />
        <xs:element name="lon_max" type="xs:float" default="180" />
    </xs:sequence>
</xs:complexType>
<xs:complexType name="RangeType">
    <xs:sequence>
        <xs:element name="min" type="xs:float" default="0" />
        <xs:element name="max" type="xs:float" default="1" />
    </xs:sequence>
</xs:complexType>
<xs:simpleType name="LocCodesType">
    <xs:restriction base="xs:string" />
</xs:simpleType>
<xs:simpleType name="LocCodeType">
    <xs:restriction base="xs:string" />
</xs:simpleType>
<xs:complexType name="MonthsType">
    <xs:sequence>
        <xs:element name="Jan" type="xs:boolean" default="true" />
        <xs:element name="Feb" type="xs:boolean" default="true" />
        <xs:element name="Mar" type="xs:boolean" default="true" />
        <xs:element name="Apr" type="xs:boolean" default="true" />
        <xs:element name="May" type="xs:boolean" default="true" />
        <xs:element name="Jun" type="xs:boolean" default="true" />
        <xs:element name="Jul" type="xs:boolean" default="true" />
        <xs:element name="Aug" type="xs:boolean" default="true" />
        <xs:element name="Sep" type="xs:boolean" default="true" />
        <xs:element name="Oct" type="xs:boolean" default="true" />
        <xs:element name="Nov" type="xs:boolean" default="true" />
        <xs:element name="Dec" type="xs:boolean" default="true" />
    </xs:sequence>
</xs:complexType>
<xs:simpleType name="NumberListType">
    <!-- should be <xs:list itemType="xs:string" /> but list are not
implemented in views -->
    <xs:restriction base="xs:string" />
</xs:simpleType>
<xs:simpleType name="ParamAbbrType">
    <xs:restriction base="xs:string" />
</xs:simpleType>
<xs:simpleType name="ParamAbbrsType">
    <xs:restriction base="xs:string" />
</xs:simpleType>
<xs:simpleType name="PositiveFloatType">
    <xs:restriction base="xs:float">
        <xs:minExclusive value="0" />
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="mlstring">
    <xs:restriction base="xs:string" />
</xs:simpleType>
<xs:complexType name="SizeFType">
    <xs:sequence>
        <xs:element name="width" type="xs:float" />
        <xs:element name="height" type="xs:float" />
    </xs:sequence>

```



```

</xs:complexType>
<xs:complexType name="SizeIType">
  <xs:sequence>
    <xs:element name="width" type="xs:int" />
    <xs:element name="height" type="xs:int" />
  </xs:sequence>
</xs:complexType>
<xs:simpleType name="ZeroToOneType">
  <xs:restriction base="xs:float">
    <xs:minInclusive value="0" />
    <xs:maxInclusive value="1" />
  </xs:restriction>
</xs:simpleType>
</xs:schema>
<xs:schema elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://datafed.net/xs/Lineage" xmlns="http://datafed.net/xs/Lineage">
  <xs:element name="LineageList" type="LineageListType" />
  <xs:complexType name="LineageListType">
    <xs:sequence>
      <xs:any namespace="##any" processContents="skip" minOccurs="0"
maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="LineageComment" type="LineageCommentType" />
  <xs:complexType name="LineageCommentType">
    <xs:sequence>
      <xs:element name="subject" type="xs:string" />
      <xs:element name="body" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="Statistics" type="StatisticsType" />
  <xs:complexType name="StatisticsType">
    <xs:sequence>
      <xs:element name="sum" type="xs:float" default="0" />
      <xs:element name="avg" type="xs:float" default="NaN" />
      <xs:element name="min" type="xs:float" default="NaN" />
      <xs:element name="max" type="xs:float" default="NaN" />
      <xs:element name="null_count" type="xs:int" default="0" />
      <xs:element name="total_count" type="xs:int" default="0" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
</types>
<message name="AddSoapIn">
  <part name="parameters" element="bgo:Add" />
</message>
<message name="SubtractSoapIn">
  <part name="parameters" element="bgo:Subtract" />
</message>
<message name="MultiplySoapIn">
  <part name="parameters" element="bgo:Multiply" />
</message>
<message name="DivideSoapIn">
  <part name="parameters" element="bgo:Divide" />
</message>
<message name="EvaluateSoapIn">
  <part name="parameters" element="bgo:Evaluate" />
</message>
<message name="ZeroGridToNaNGridSoapIn">
  <part name="parameters" element="bgo:ZeroGridToNaNGrid" />
</message>
<message name="MakeVector2DSoapIn">
  <part name="parameters" element="bgo:MakeVector2D" />
</message>
<message name="SoapOut">
  <part name="parameters" element="bgo:Response" />
</message>
<portType name="MapGridOperatorSoap">
  <operation name="Add">

```

```

        <input message="bgo:AddSoapIn" />
        <output message="bgo:SoapOut" />
    </operation>
    <operation name="Subtract">
        <input message="bgo:SubtractSoapIn" />
        <output message="bgo:SoapOut" />
    </operation>
    <operation name="Multiply">
        <input message="bgo:MultiplySoapIn" />
        <output message="bgo:SoapOut" />
    </operation>
    <operation name="Divide">
        <input message="bgo:DivideSoapIn" />
        <output message="bgo:SoapOut" />
    </operation>
    <operation name="Evaluate">
        <input message="bgo:EvaluateSoapIn" />
        <output message="bgo:SoapOut" />
    </operation>
    <operation name="ZeroGridToNaNGrid">
        <input message="bgo:ZeroGridToNaNGridSoapIn" />
        <output message="bgo:SoapOut" />
    </operation>
    <operation name="MakeVector2D">
        <input message="bgo:MakeVector2DSoapIn" />
        <output message="bgo:SoapOut" />
    </operation>
</portType>
<binding name="MapGridOperatorSoap" type="bgo:MapGridOperatorSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"
/>
    <operation name="Add">
        <documentation>Adds two grids.</documentation>
        <soap:operation soapAction="http://datafed.net/MapGridOperator/Add"
style="document" />
        <input>
            <soap:body use="literal" />
        </input>
        <output>
            <soap:body use="literal" />
        </output>
    </operation>
    <operation name="Subtract">
        <documentation>Subtracts two grids.</documentation>
        <soap:operation soapAction="http://datafed.net/MapGridOperator/Subtract"
style="document" />
        <input>
            <soap:body use="literal" />
        </input>
        <output>
            <soap:body use="literal" />
        </output>
    </operation>
    <operation name="Multiply">
        <documentation>Multiplies two grids.</documentation>
        <soap:operation soapAction="http://datafed.net/MapGridOperator/Multiply"
style="document" />
        <input>
            <soap:body use="literal" />
        </input>
        <output>
            <soap:body use="literal" />
        </output>
    </operation>
    <operation name="Divide">
        <documentation>Divides two grids.</documentation>
        <soap:operation soapAction="http://datafed.net/MapGridOperator/Divide"
style="document" />
        <input>
            <soap:body use="literal" />
        </input>

```

```

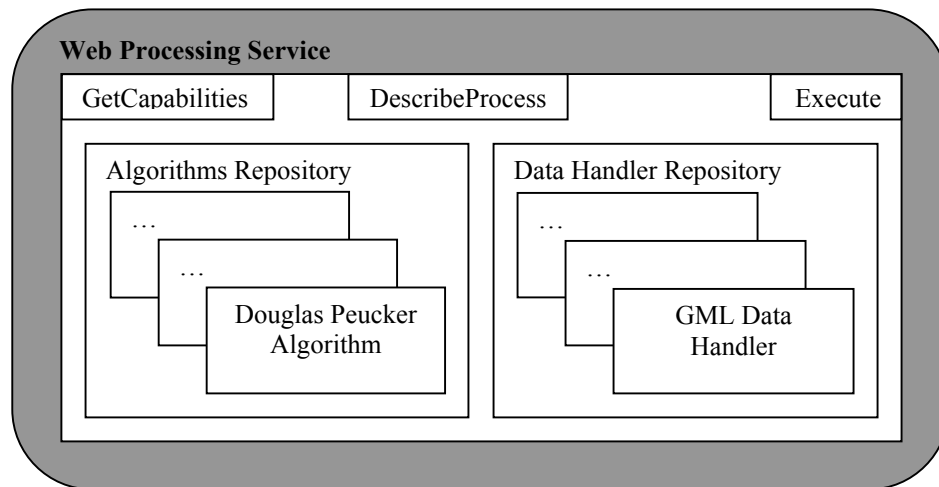
        <output>
          <soap:body use="literal" />
        </output>
      </operation>
      <operation name="Evaluate">
        <documentation>Evaluates an expression between corresponding cells in two
        grids.</documentation>
        <soap:operation soapAction="http://datafed.net/MapGridOperator/Evaluate"
        style="document" />
        <input>
          <soap:body use="literal" />
        </input>
        <output>
          <soap:body use="literal" />
        </output>
      </operation>
      <operation name="ZeroGridToNaNGrid">
        <documentation>If there are no non-zero values in this grid, all the values
        will be set to zero.</documentation>
        <soap:operation
        soapAction="http://datafed.net/MapGridOperator/ZeroGridToNaNGrid" style="document" />
        <input>
          <soap:body use="literal" />
        </input>
        <output>
          <soap:body use="literal" />
        </output>
      </operation>
      <operation name="MakeVector2D">
        <documentation>Makes one 2D vector grid out of two grids.</documentation>
        <soap:operation soapAction="http://datafed.net/MapGridOperator/MakeVector2D"
        style="document" />
        <input>
          <soap:body use="literal" />
        </input>
        <output>
          <soap:body use="literal" />
        </output>
      </operation>
    </binding>
    <service name="MapGridOperator">
      <port name="MapGridOperatorSoap" binding="bgo:MapGridOperatorSoap">
        <soap:address location="http://webapps.datafed.net/MapGridOperator.asmx" />
      </port>
    </service>
  </definitions>

```

## Annex D WPS best practice implementation

### D.1 Architecture overview

This Annex describes a possible software design for a pluggable processing framework, as it has been introduced in [13]. The framework has to be pluggable for the different processes and for the different data handlers (e.g. for GML2). This ensures that theoretically each process can make use of each data handler. Thus each process can be feed with all available data handlers. The process by itself can be deployed on-the-fly in the architecture and thereby be made available to the potential web-clients. Figure 11 illustrates this architectural approach.



**Figure 11: Pluggable architecture for WPS**

The suggested architecture has been implemented in a WPS prototype, which is hosted as an incubator project at 52North.

## Annex E WPS XML examples

### E.1 WPS DescribeProcess example for Douglas Peucker simplification

```

<?xml version="1.0" encoding="UTF-8"?>
<wps:ProcessDescriptions
xmlns:wps="http://www.opengeospatial.net/wps"><wps:ProcessDescription
processVersion="2" storeSupported="true" statusSupported="false"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ows="http://www.opengeospatial.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink">

  <ows:Identifier>org.n52.wps.server.algorithm.simplify.DouglasPeuckerA
  lgorithm</ows:Identifier>
  <ows:Title>douglasPeucker algorithm</ows:Title>
  <ows:Abstract>Uses JTS implementation. Does not support topological
  awareness</ows:Abstract>
  <ows:Metadata xlink:title="spatial"/>
  <ows:Metadata xlink:title="geometry"/>
  <ows:Metadata xlink:title="douglas peucker"/>
  <ows:Metadata xlink:title="GML"/>
  <wps:DataInputs>
    <wps:Input>
      <ows:Identifier>FEATURES</ows:Identifier>
      <ows:Title>input features</ows:Title>
      <ows:Abstract>Just features</ows:Abstract>
      <wps:ComplexData defaultFormat="text/XML"
  defaultSchema="http://www.opengeospatial.org/gmlpacket.xsd">
        <wps:SupportedComplexData>

          <wps:Schema>http://schemas.opengis.net/gml/2.1.2/feature.xsd</wps:Sch
  ema>

          </wps:SupportedComplexData>
          </wps:ComplexData>
          <wps:MinimumOccurs>1</wps:MinimumOccurs>
        </wps:Input>
      <wps:Input>
        <ows:Identifier>TOLERANCE</ows:Identifier>
        <ows:Title>Tolerance Value for DP Alg</ows:Title>
        <ows:Abstract/>
        <wps:LiteralData>
          <ows:DataType ows:reference="xs:double"/>
          <ows:AllowedValues>
            <ows:Value/>
          </ows:AllowedValues>
        </wps:LiteralData>
        <wps:MinimumOccurs>1</wps:MinimumOccurs>
      </wps:Input>
    </wps:DataInputs>
    <wps:ProcessOutputs>
      <wps:Output>
        <ows:Identifier>SIMPLIFIED_FEATURES</ows:Identifier>
        <ows:Title>smooth geometries</ows:Title>

```

```
        <ows:Abstract>GML stream describing the smooth
feature.</ows:Abstract>
        <wps:ComplexOutput defaultFormat="text/XML"
defaultSchema="http://www.opengeospatial.org/gmlpacket.xsd">
        <wps:SupportedComplexData>

        <wps:Schema>http://schemas.opengis.net/gml/2.1.2/feature.xsd</wps:Sch
ema>
        </wps:SupportedComplexData>
        </wps:ComplexOutput>
        </wps:Output>
        </wps:ProcessOutputs>
</wps:ProcessDescription></wps:ProcessDescriptions>
```

## Annex F Annex F OWS-4 Earth Observation Demo

A demonstration scenario focused on earth observations emerged from the SWE thread with the objective of processing imagery. The scenario involves the following sequence and is illustrated in the diagram below.

1. Analyst searches the NASA Earth Science Gateway (ESG) for sea-level pressure forecasts and finds the GEOS-5 model is accessible via WMS and WCS.
2. Analyst retrieves forecasts for sea-level pressure from the GEOS-5 model via the NASA WCS. When the pressure falls below a threshold (calculated via a WPS), the analyst determines the potential for hurricane landfall is likely and seeks satellite imagery.
3. Analyst searches NASA ESG for recent images of the landfall area. None are found.
4. The EO-1 is tasked for imagery over the landfall area for the earliest possible date, for the date of landfall, and for one day after landfall using the Vigtel SPS.
5. The SPS notifies the analyst of the location of the collected images
6. The analyst is interested in viewing flooded areas of the images. He searches the NASA ESG for services providing EO-1 flood algorithm services. He finds a GMU BPEL workflow for calculating EO-1 flood images.
7. He submits a WCS request for a GeoTiff that shows flooded areas. The WCS request triggers the GMU BPEL workflow at GMU to calculate the ratio of two EO-1 bands which provides an indication of flooded areas. The ratio is calculated using the WashU Binary Grid WPS. The calculated grid is returned as GeoTiff.
8. The GeoTiff is viewed in a WMS client.

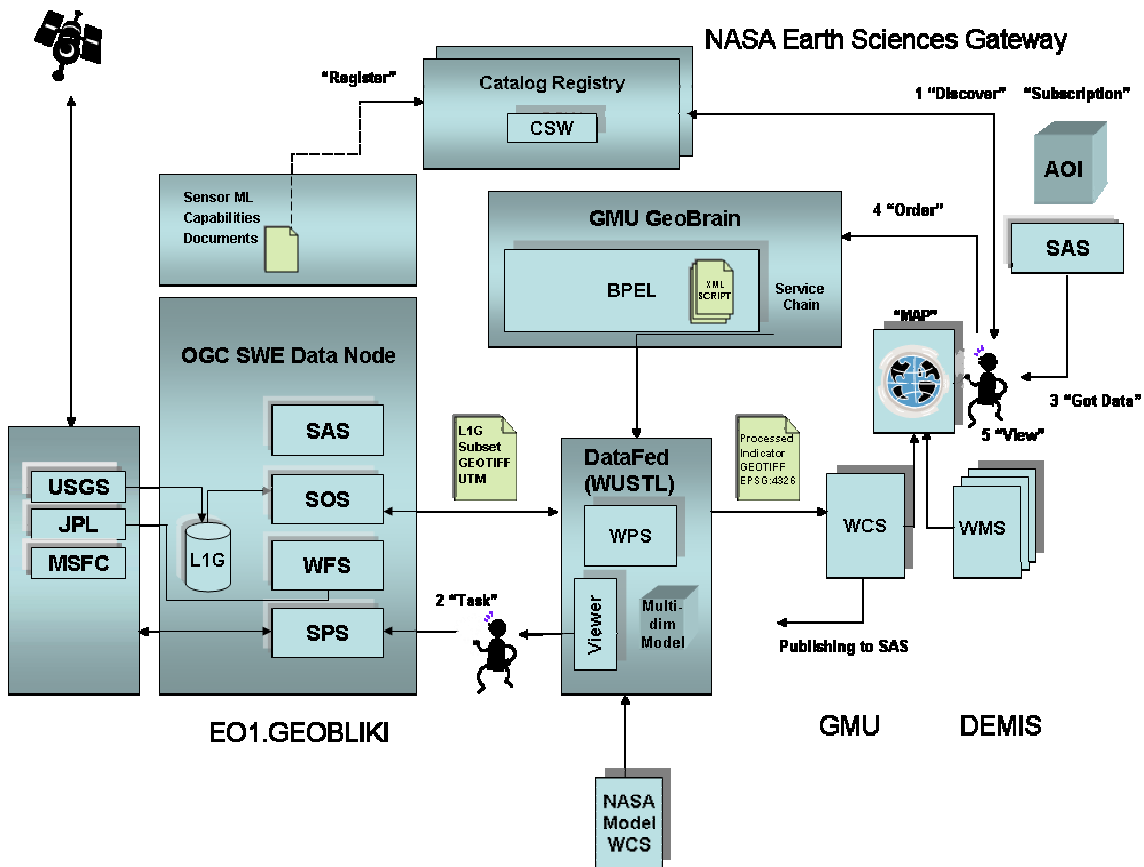


Figure 12: Components and services in the OWS-4 Earth Observation demo.

In the service flow, the WPS is a binary grid processing service. The binary operator service takes two grids as inputs, performs a calculation on them, and generates an output grid.

In future revisions to the earth observation demonstration, a more involved BPEL workflow involving WPSes is expected to include processing the EO-1 image through a WCTS before passing it through the binary processing service.



## Bibliography

- [1] OWS Common Implementation Specification [OGC 05-008c1], 2005-11-22,  
[https://portal.opengeospatial.org/files/?artifact\\_id=8798](https://portal.opengeospatial.org/files/?artifact_id=8798)
- [2] Web Processing Service (WPS) [OGC 05-007r4]
- [3] Topic 12 - The OpenGIS Service Architecture [OGC 02-112], 2001-09-14,  
[http://portal.opengeospatial.org/files/?artifact\\_id=1221](http://portal.opengeospatial.org/files/?artifact_id=1221)
- [4] Establishing an OGC Web Processing Service for generalization processes, Theodor Foerster & Jantien Stoter, June 25th 2006,  
<https://portal.opengeospatial.org/wiki/twiki/bin/view/OWS4/GeneralizationProcessingService>
- [5] WPS Change Request - Support POST in ComplexValueReference [06-151], Steven Keens, 2006-11-03,  
[http://portal.opengeospatial.org/files/?artifact\\_id=18557&version=1](http://portal.opengeospatial.org/files/?artifact_id=18557&version=1)
- [6] OGC O4-078, OWS2 IH4DS Service Chaining IPR
- [7] OWS-2 Service Chaining, John Vincent, David Rosinger
- [8] BPEL for Image Handling IPR
- [9] OWS-3 Imagery Workflow Discussion Paper
- [10] OWS-3 Imagery Workflow IPR
- [11] SWE WPS Description, Stefan Falke
- [12] Galdos Issue Report for OWS-4,  
<https://portal.opengeospatial.org/wiki/twiki/pub/OWS4/ClippingIssueInWFS/DescriptionofGPWclippingissue.pdf>
- [13] Foerster, T. An open software framework for Web Service-based geo-processes *Free and Open Source Software for Geoinformatics (Foss4g 2006)*, **2006**
- [14] WSRF - Defining an open framework for modelling and accessing state full resources using Web services",  
[http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wsrf](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf)