# Open Geospatial Consortium Inc.

Date: 2007-01-25

Reference number of this OpenGIS® IP initiative document: **OGC 06-187r1**

Version: **0.0.9**

Category: OGC® Public Discussion Paper

Editor: Steven Keens

# OWS-4 Workflow IPR

## Workflow descriptions and lessons learned

| | |
|---|---|
| Document type: | OpenGIS® Discussion Paper |
| Document subtype: | OGC Interoperability Program report |
| Document stage: | Proposed version 0.0.9 |
| Document language: | English |

## Contents

## Figures Page

## Tables Page

## 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

George Mason University

Washington University in St. Louis

Spacebel

International University Bremen

## iii. Document contributor contact points

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

| Contact | Company | Address | Phone | Email |
|---|---|---|---|---|
| Steven Keens | PCI Geomatics | 490 St Joseph Blvd. Suite 400, Gatineau, Quebec, J8Y 3Y7, Canada | 819-770-0022 x204 | skeens@pcigeomatics.com |
| Peisheng Zhao | George Mason University | 9801 Greenbelt Road Suite 316-7 Lanham 20706 USA | 301-220-1536 | pzhao@gmu.edu |
| Stefan Falke | Washington University in St. Louis | Campus Box 1180 1 Brookings Drive St. Louis, MO 63130 USA | 314-935-6099 | stefan@wustl.edu |
| The Hoa Nguyen | Spacebel | Belgium | +32 2 658 20 11 | TheHoa.Nguyen@spacebel.be |
| Peter Baumann | International University | Gemany | +49 421 200 3178 | p.baumann@iu-bremen.de |

| | | | | |
|---|---|---|---|---|
| | Bremen, GmbH | | | |
| | | | | |
| | | | | |

## iv. Revision history

| Date | Release | Editor | Primary clauses modified | Description |
|---|---|---|---|---|
| December 11, 2006 | 0.0.1 | Steven Keens | All | Major sections |
| January 11, 2007 | 0.5.0 | Steven Keens | All | Incorporated sections provided by Peisheng Zhao and Stefan Falke |
| January 18, 2007 | 0.9.0 | Steven Keens | 8, appendices, and various others. | Incorporated sections provided by The Hoa Nguyen. |
| 5/1/07 | 0.0.9 | Carl Reed | Various | Get document ready for posting as a DP |
| | | | | |
| | | | | |

## v. Foreword

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.

This work is an interoperability program report compiled by the OGC Web Services, Phase 4 initiative of the OGC Interoperability Program.

This document presents the work completed with respect to the workflows within the OWS-4.

This is not a normative document.

# OWS-4 Workflow IPR

# 1 Introduction

The OGC's Web Services Interoperability Program phase 4 (OWS-4) included a requirement to implement workflows for solving various problems. This document examines five workflows discussed during the course of the OWS-4 project. It introduces the workflows and it describes why they were needed, the final design, some lessons learned, and some possible future directions.

The workflows originate from two OWS-4 threads: Geo-Processing Workflow (GPW) and Sensor Web Enablement (SWE). The GPW thread dealt mainly with workflows related to processing features while the SWE thread processed imagery from sensors and coverages. All workflows used various OGC services such as Web Coverage Service (WCS), Web Feature Service (WFS), and Web Coordinate Transformation Service (WCTS). Some non-OGC services standards like WS-Addressing were also used to accomplish the tasks.

## 1.1 Adapter Pattern

To reduce the effort required to implement the workflows and to simplify the task of accessing workflows commonly wrap the workflow with a well know OGC service. For example, the data reduction workflow exposes the underlying process as a WFS. This is a recurring design solution for OWS-4 workflows. It is commonly referred to as the Adapter Pattern (also known as a wrapper). An adapter allows services to work together that normally could not because of incompatible interfaces by wrapping an already existing service with another service interface. A good description of the adapter pattern at the site: http://en.wikipedia.org/wiki/Adapter_pattern.

Several terms have been used in this document and during discussions within the OWS-4 groups to identify the adapter service, such as: front end, virtual service, proxy, and composite.

The main advantage of the adapter pattern is that existing clients can be used, thus improving interoperability. The main disadvantage is that it is difficult to expose new functionality because it has to be grafted onto the interface thus, requiring modifications to existing clients.

### 1.1.1 Possible alternative to the adapter pattern

An alternative to the adapter pattern might be to use an interface that is already highly malleable. An example of a malleable interface is the Web Processing Service (WPS). WPS is a simple interface specification and will soon be an adopted OGC standard. Exposing a workflow as a WPS process could be a viable solution for rapid development of workflows. A generic WPS client can be used initially to execute the workflow/process. Later, dedicated clients can be used if the workflow/process proves to be valuable, eventually becoming a WPS profile.

# 2 Data reduction workflow

## 2.1 Introduction

One item of the GPW was to set up a workflow to serve data through a network with limited bandwidth capacities. That workflow was originally referred to as the Clip-Zip-Ship Service in the OWS-4 RFQ.

Due to the nature of the use case the quantity of data had to be reduced by clipping the features and then transferring them as a zipped archive. Generalization was added to the scenario to further reduce the data to eliminate imperceptible or unimportant aspects of the data.

## 2.2 Requirements

The GPW group agreed that this workflow would have the following characteristics:

- Implement as a WFS adapter with the workflow performing the work behind the scenes. The operations would be opaque to the client in the sense that the data reduction parameters will be fixed. Although described as a hack, this approach minimizes impact on OWS-4 clients and some participants feel there is value in this approach for the real world.

- One purpose of this workflow is to deliver data to clients with limited bandwidth and storage. This is comparable to web sites designed to be accessible by mobile devices.

- A second purpose of this workflow is to deliver data to a client suitable for use as part of a base map. This data will be used for reference purposes only, and is not intended to be edited.

- During the discussions we discussed whether the base map should be delivered as maps via a WMS or as GML via a WFS. Although there are use cases for this approach, the sponsors indicated their intent was that the workflow delivers feature data (GML) to the client.

- Another requirement for OWS-4 GPW was to allow the light weight client to edit features on the light weight client. Due to the fact that the data reduction workflow returns incomplete features the light weight WFS client can not edit the reduced features. This requirement was not met during OWS-4.

## 2.3 Use cases

### 2.3.1 Retrieve reduced feature data

The following steps form the data reduction work flow implemented in the OWS-4:

1. User changes area of interest on the map
2. Client sends request to a WFS adapter server requesting data for base map on that AOI.
3. WFS adapter server invokes BPEL script to perform data reduction. BPEL script performs following steps:
    1. Retrieves full GML document from main WFS using a filter with specified AOI
    2. Run generalization process on the resulting GML.
    3. Run clipping process on the results of the generalization.
    4. Retrieves clipped results and returns them to proxy server.

   It might be possible to optimize the order of steps 2 to 4 to first perform clipping then generalization but that depends upon the actual data. The implemented workflow followed the above use case.

4. Proxy server returns clipped results to client. HTTP supports GZIP encoding so no specific step is required to actually zip the data. This should be supported by the HTTP client and the HTTP server running the WFS.

### 2.3.2 User modifies reduced features

This use case was not implemented but it was discussed during the OWS-4 GPW work.

1. Assume that the user has the latest base map – i.e. has recently retrieved the reduced data set for the current area of interest.
2. User selects one or more features needing edits.
3. Client downloads complete data for all features needing edits.
4. User edits features as required and informs client when finished.
5. Client submits edited features via a WFS transaction. This is where the Feature update workflow is called, see clause 3.

## 2.4 Design

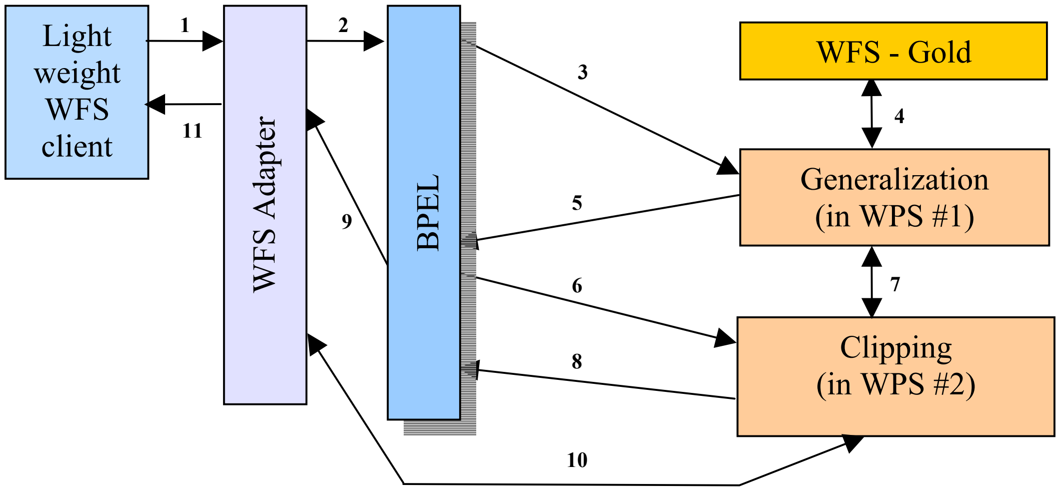

**Figure 1: Data reduction workflow sequence**

Figure 1 shows the sequence that services are activated within the data reduction workflow. The step numbers are described in Table 1.

**Table 1: 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 |

### 2.4.1 BPEL Diagram

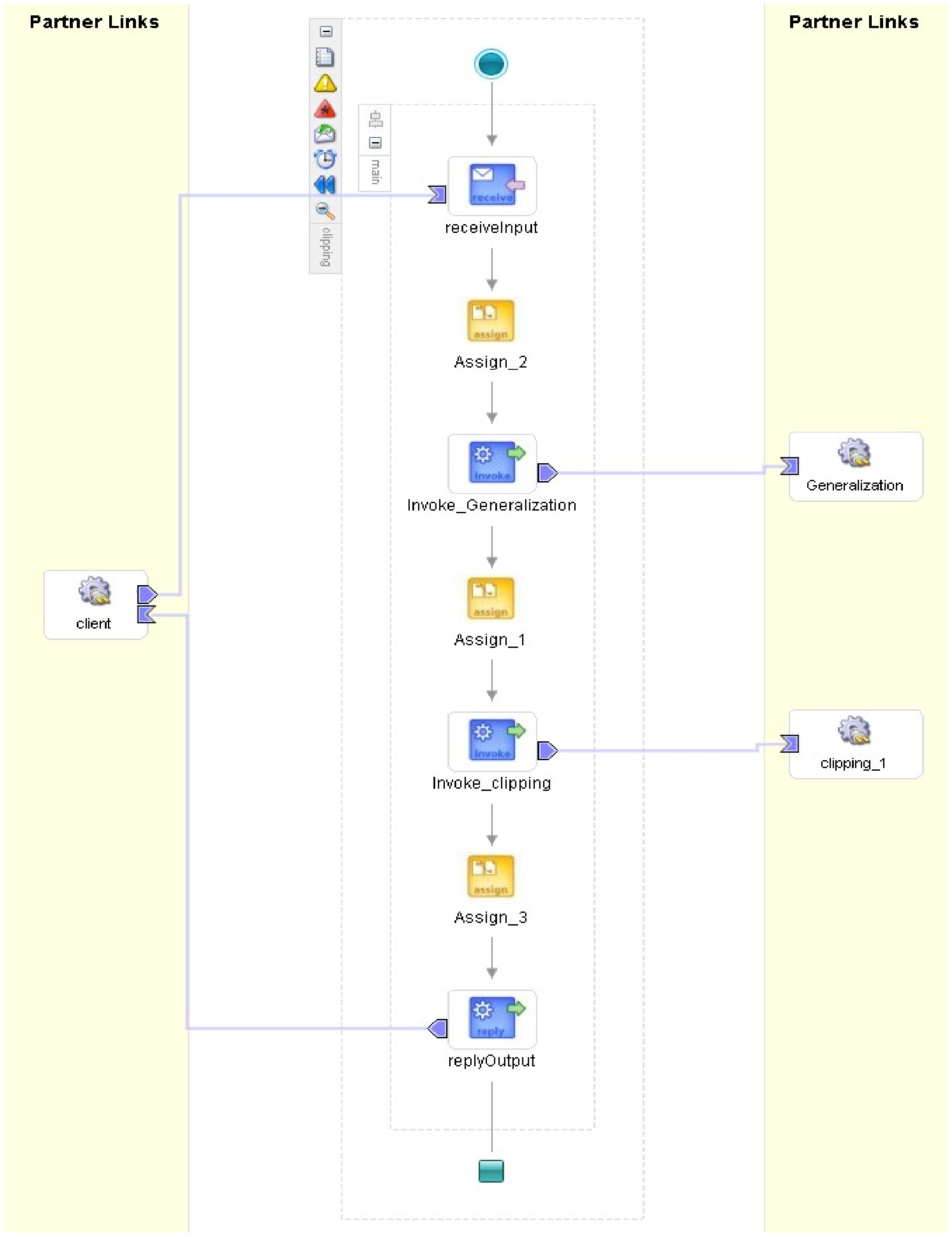

**Figure 2: BPEL diagram for data reduction workflow**

## 2.5 Services and Components

### 2.5.1 WFS-Gold

The WFS-Gold is the ultimate data source end point. All feature data comes from this WFS server.

Service URL: http://geoserver.itc.nl:8080/geoserver/wfs

### 2.5.2 WPS Generalization Process

The Douglas-Peucker generalization algorithm was used in the data reduction workflow. It was implemented by Theodore Foerster at ITC as a WPS process. The implementation is detailed in [8] subclause 6.2.

Service URL: http://geoserver.itc.nl:8080/wps/WebProcessingService

To test the Generalization WPS process please visit the test page at: http://geoserver.itc.nl:8080/wps/test.html

### 2.5.3 WPS Clipping Process

The clipping service was implemented by GMU as a WPS process. The implementation is detailed in [8] subclause 6.3. Links to the clipping process are available from that same location.

Service URL: http://geobrain.laits.gmu.edu:8099/wps/WebProcessingService

### 2.5.4 WFS Adapter

WFS adapter is implemented by GMU as a WFS to perform workflow behind scene just using WFS request.

Service URL: http://geobrain.laits.gmu.edu:8099/owswrapper/wfs100

## 2.6 Lessons Learned

The design of the data reduction workflow (see subclause 2.4) is nearly the optimal method. It is possible to implement the workflow using a naïve approach whereby all intermediate results are transmitted to and from the BPEL engine. This can be costly in both resources and time when the intermediate results are large. Instead the BPEL script was implemented such that references are passed wherever possible.

This workflow lends itself well to the adapter pattern with a WFS front end. All WFS clients can remain ignorant of the fact that a workflow does all of the work. However, the light weight client used in this workflow had a problem

# 3 Feature update workflow

## 3.1 Introduction

One item of the GPW was to create a workflow to perform topological quality assessment tests on feature transactions before applying them to the database. This workflow is started when the use case "User modifies reduced features" described in clause 2.3.2.

## 3.2 Requirements

- Light weight client with limited bandwidth and resources sends a WFS transaction to a WFS-T
- Transaction results need to be tested for topological quality before being applied to database.
- Topological assessment must run on all features to ensure topological validity.

These requirements lead to the idea there be two databases:

1. A gold database, which contains only valid features.
2. A silver database where transactions are first performed and tests are performed against it.

When the topological assessment finishes and is successful then, and only then is the transaction applied to the gold database.

## 3.3 Design

Two designs were considered: synchronous and asynchronous versions. Again, the synchronous design was chosen because it was simpler to implement. Both designs are described below but the synchronous implementation is further elaborated.

### 3.3.1 Synchronous

The feature update workflow ingests WFS transactions. A WFS transaction passes the features through a topological quality assessment service where they are evaluated. The design is as Figure 3 which requires two databases:

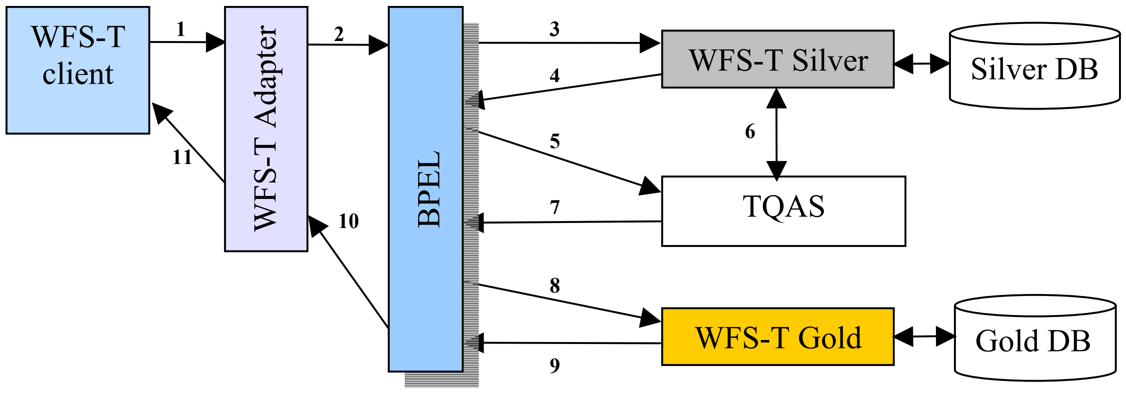

**Figure 3: Synchronous feature update workflow**

**Table 2: Synchronous feature update workflow**

| Step | Description | Interface |
|---|---|---|
| 1 | Client sends 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 | |

### 3.3.2 Asynchronous

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

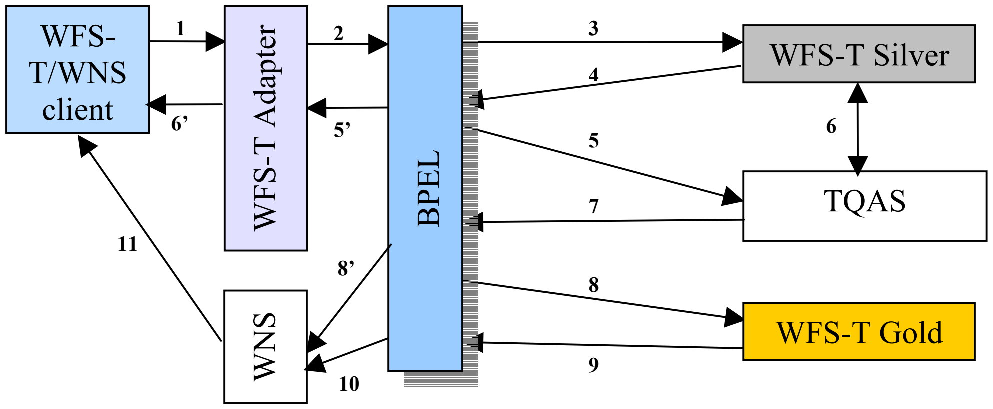

**Figure 4: Asynchronous Feature update workflow**

**Table 3: 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 | |

## 3.4 Services and Components

The WFS-T Gold and the WFS-T Silver can be implemented as two separate servers or one server with two feature types. The only requirement is that the application schemas be exactly the same for the silver and gold feature types.

## 3.5 TQAS Workflow

The feature update workflow uses the TQAS workflow to perform the topological quality assessment.

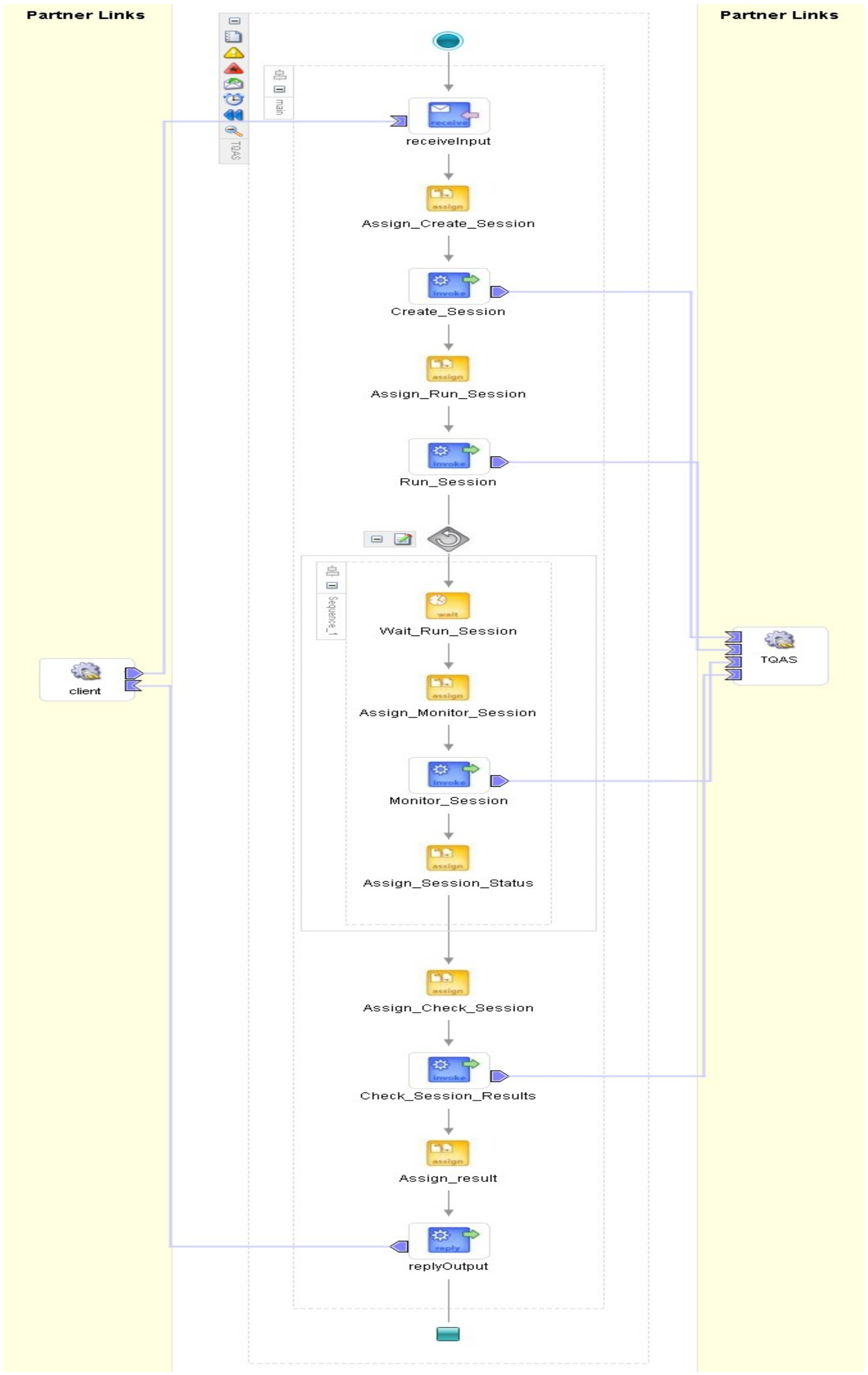

**Figure 5: TQAS BPEL diagram**

In the feature update workflow design, when the topological assessment finishes and is successful then, and only then, is the transaction applied to the gold database. However, 1Spatial actually wraps all functions of one software suite into the Topology Quality Assessment Service (TQAS). Thus, the TQAS includes too many service endpoints and relevant operations so that it can not be used easily as a web service if not an explicit document. The document provided by 1Spatial (http://portal.opengeospatial.org/files/?artifact_id=18832) only involves one of the TQAS service endpoints – the SessionManagerEndPoint without any information about accessing WFS-T. Based on this document, GMU implemented the TQAS workflow only invoking the SessionManagerEndPoint as shown in Figure 6. This workflow requires more complex control, such as *wait* and *while*.

GMU implemented a WFS-T adapter, but it was not used in the demo due to lack of availability of the silver and gold WFS-T servers.

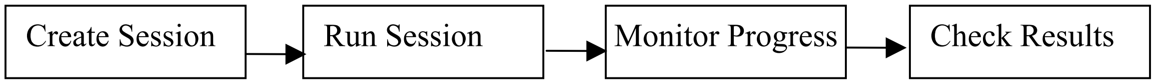

**Figure 6: TQAS workflow sequence**

**Table 4: TQAS workflow**

| Step | Description | Interface |
|---|---|---|
| 1 | Create a session with known data store reference and checking rule reference. If it succeeds, it returns a session id. | Create SOAP request |
| 2 | Use the session id run session. | run SOAP request. |
| 3 | Use the session id to monitor the session progress periodically until it return “finished”. | getProgress SOAP request |
| 4 | Obtain the session results: the summary which given overall conformance levels for the rules checked and the detailed per-feature metadata. | getResults SOAP request |

## 3.6 Lessons Learned

The simplicity is the main goal of Web service design. Don’t include too many ports and relevant operations in one service. Otherwise it can not be used easily without a good document.

BPEL is good enough to represent complex control structures, such as *wait* and *while* in TQAS workflow.

The design using two databases can cause some synchronization issues. For example, when two transactions occur concurrently how does the topological assessment handle that? If transaction A produces some invalid features and transaction B produces valid features, transaction A would have to be rolled back before transaction B can be applied and tested.

This leads to the idea that each transaction should have its own copy of the database. That is unreasonable since the database can be extremely large. Another solution is that the topological assessment be able to restrict its assessment to those features that are modified by the transaction. Again, that could prove to be expensive and maybe impossible.

### 3.6.1 Future directions - asynchronous transactions

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

The OGC needs to start addressing the asynchronous requests issue. It's a subject that has appeared many times in past projects and is often pushed aside for the simpler synchronous solution. However, the simpler synchronous solution has its own drawbacks that we can not surmount. Future projects need to explicitly require asynchronous solutions. Some possible starting points are WS-Addressing or the WNS. Note: WS-Addressing has been successfully used in the ESA SSE Workflow, see clause 5.

With the current WFS specification it is impossible for a client to undo a previous successful transaction. For example, how should the workflow deal with the case when the TQAS fails? The transaction on the Silver WFS needs to be undone. How can the workflow undo it? There are two options: copy the gold WFS or rollback the transaction. Copying is an expensive task and can not be used in an environment with many users. Rollback would be ideal but it is not available within the WFS. The lack of rollbacks in the WFS makes it very expensive to implement the feature update workflow.

Transactions that hit several services at once are impossible to implement with the current crop of OGC standards. This issue will arise more often as the OGC attempts to integrate services in to workflows. Most such issues have already been solved by the general software community. Current theory includes ideas like optimistic or pessimistic concurrency controls and ACID transactions, and replication. These theories may need to be introduced into the OGC's specifications to allow for externally controlled transactions. One approach would be to create profiles. Another approach might be to put the external transaction control interface into a separate document that applies to all OGC services.

# 4 OWS-4 Earth Observation (EO1) Demo Workflow

## 4.1 Introduction

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.

## 4.2 Design

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 Vightel 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 that uses a WCS interface.

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.

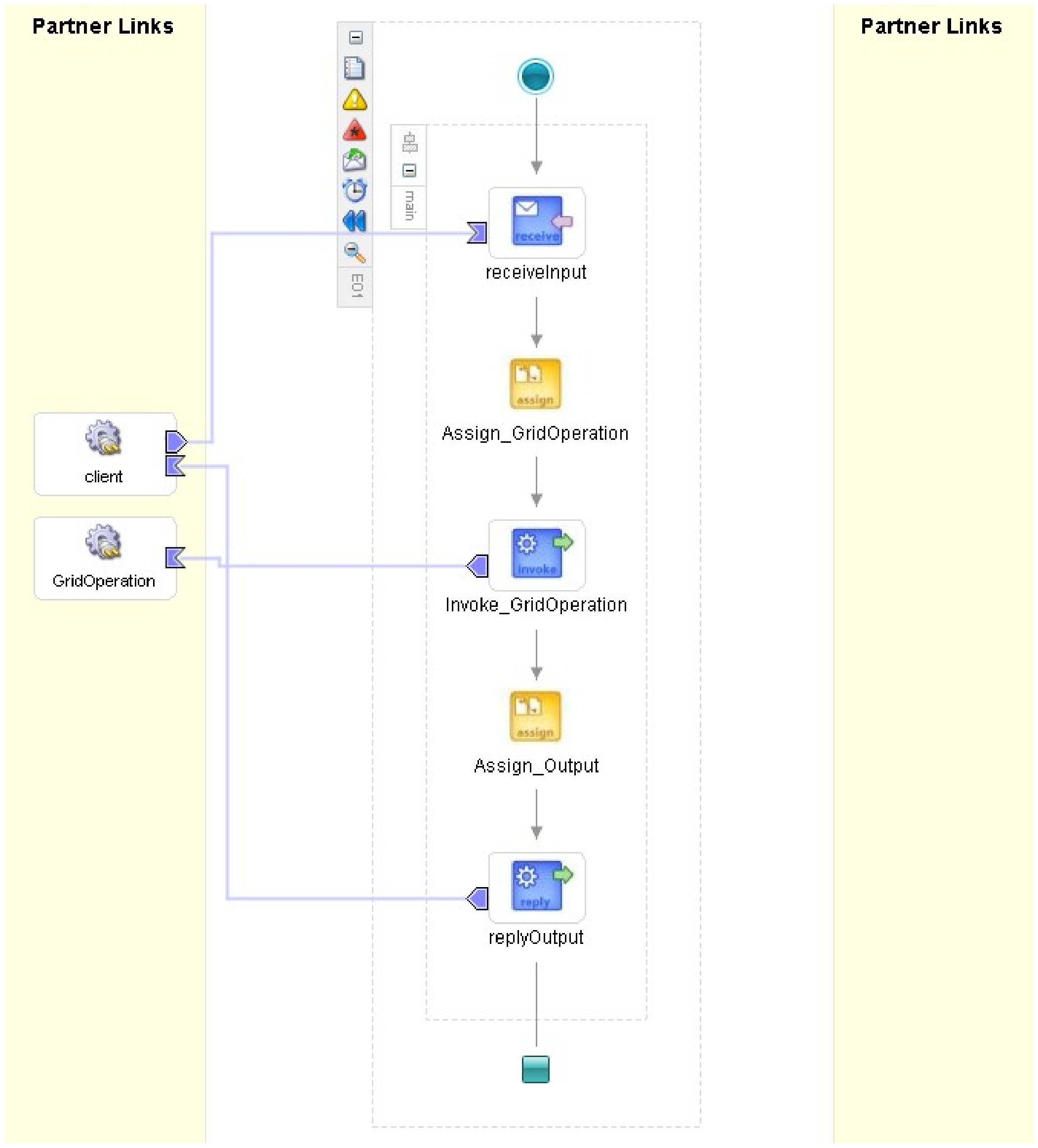

**Figure 7: BPEL diagram for EO1 demo**

## 4.3 Services and Components

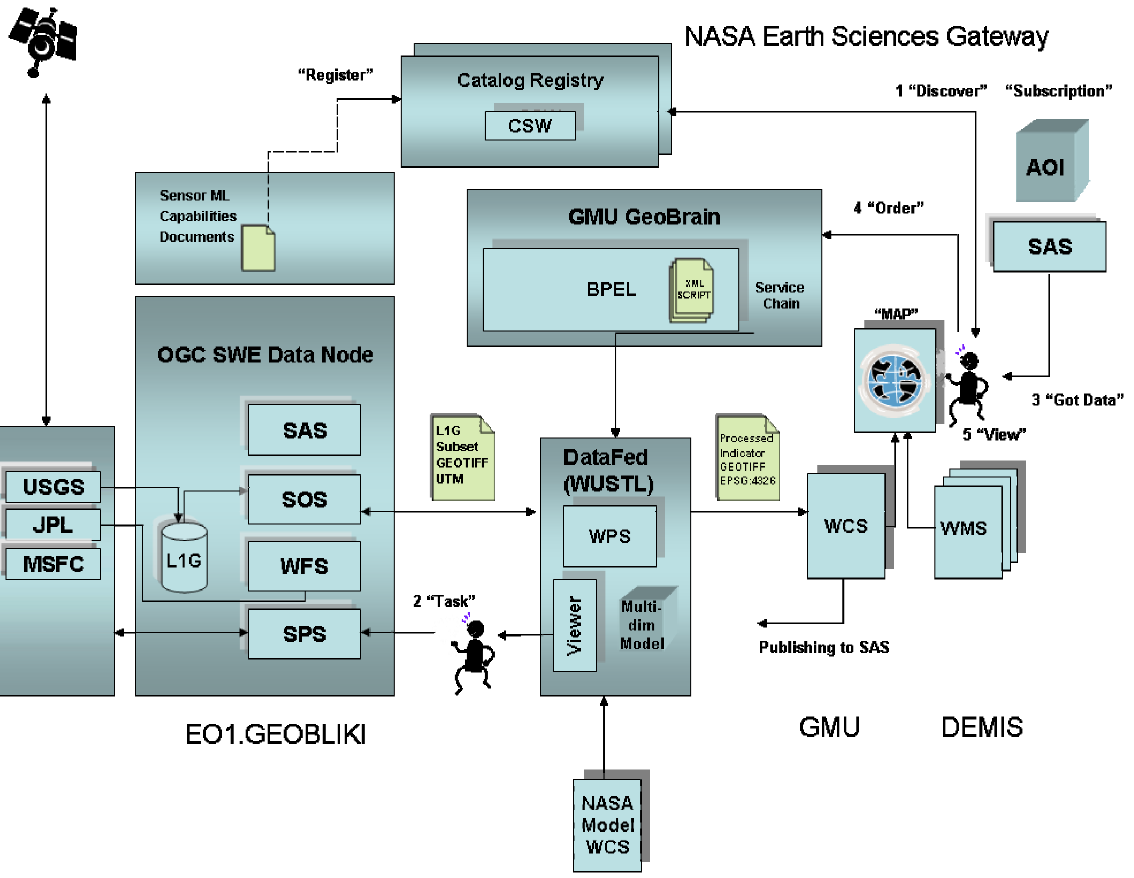

**Figure 8: Components and services in the OWS-4 Earth Observation demo**

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.

## 4.4 Lessons Learned

Most of OGC services, such as SOS and SPS, have no WSDL files. Therefore it is very difficult integrate these services into workflow. To solve this issue, the HTTP Get requests of these services are always used as other service inputs directly.

## 4.5 Future directions

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

# 5 ESA SSE Workflow

## 5.1 Introduction

This section is a summary of [8] which contains all of the details related to the European Space Agency's (ESA) Service Support Environment (ESA) workflow. See sections 2, 5, 6.1, and 9-Annex of that document.

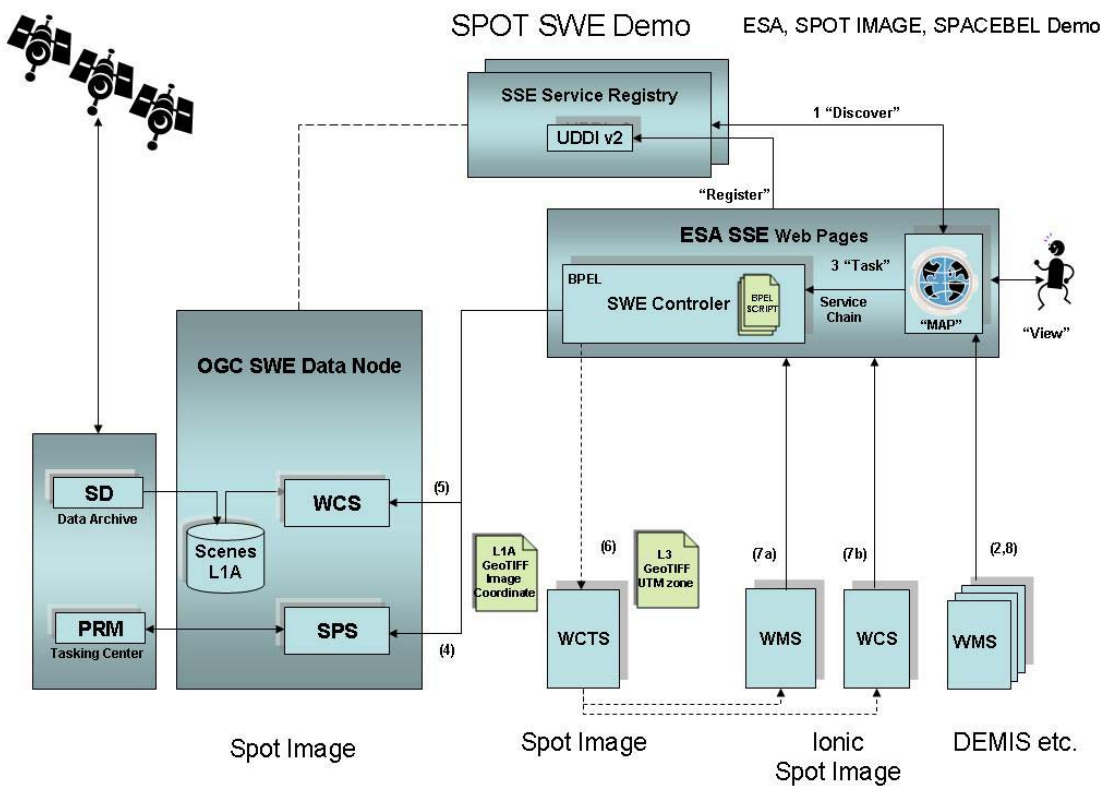

Figure 9: OWS-4 SWE Demo Scenario

The SWE Controller component is designed as a composite service made by the ground segment services SPS, WCS, WCTS and WMS. The composite service is not different to another SPS service in the interfaces, but in the added-value it brings to. For example, the Spot Image SPS service can only provide the user the raw output image level 1A. With the SWE controller composite service, thanks to the WCTS, users can choose to receive the output image as raw image level or Otho image:

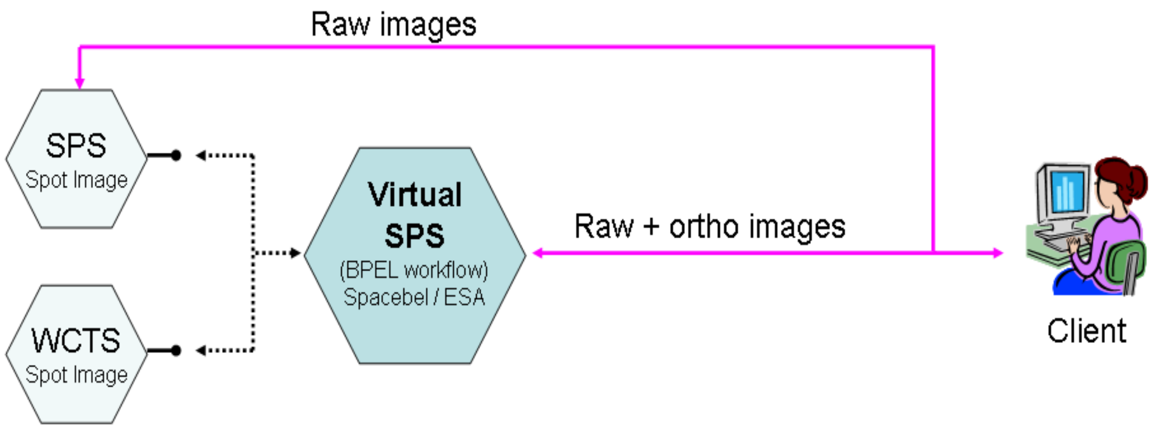

**Figure 10: The virtual SPS is a composite service**

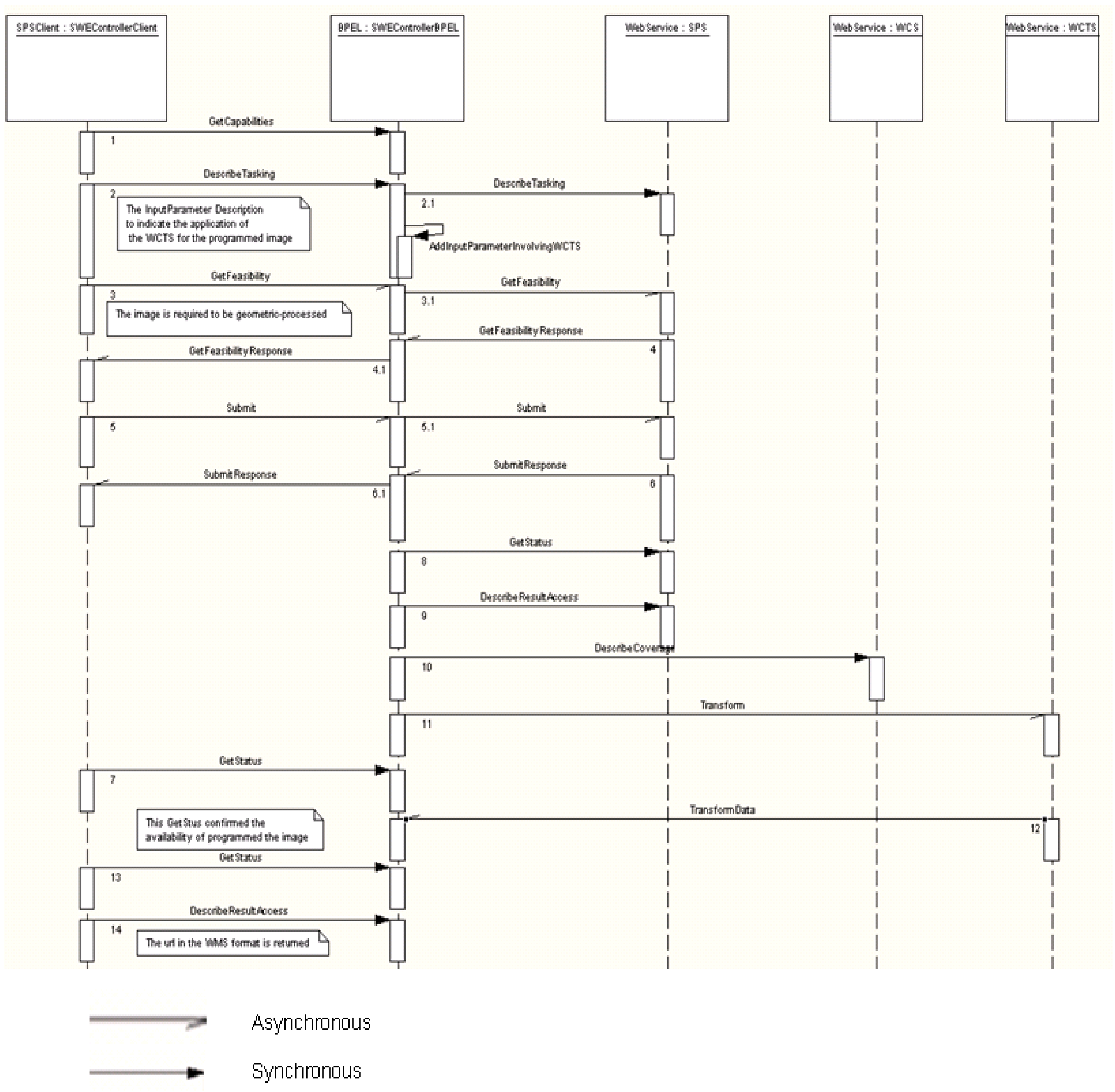

**Figure 11: Web Services interactions**

## 5.2 Lessons Learned

Most of OGC services, such as SPS and WCTS, have business operations that may require a long time to process. To alleviate the problems related to long running process the test bed successfully applied the WS-Addressing standards to integrate those services into a value-added service chain. Thanks to the WS-addressing, an operation can be designed to be synchronous or asynchronous without having to change the input or the request and the output or the response.

The test bed successfully integrated different OGC service interfaces, such as HTTP GET/Post and SOAP/HTTP, into the same service chain. This shows that it is possible to build value-added service chains from HTTP GET/POST OGC services and other OGC SOAP/HTTP services. For example, in the test bed, the WCS service is accessible via HTTP Get, while the SPS and WCTS are accessible on SOAP/HTTP.

# 6 Web Coverage Processing Service (WCPS)

## 6.1 Introduction

In response to several requests to the WCS Revision Working Group to incorporate more advanced (i.e., analysis) functionality into WCS, International University Bremen (IUB) had already started development of a Web Coverage Processing Service (WCPS) at the end of 2005. In OWS-4, WCPS is being evolved further in both specification and implementation, as well as presented, discussed, and continuously technically improved. As this work is done as a fully in-kind contribution, there are no ample resources available.

WCPS offers a coverage language which allows phrasing of arbitrary complex operations by functional composition. Requests can process one or more coverages in sequence, and can additionally combine any number of coverages (for example masking, overlaying). WCPS requests are composed by the client and shipped to the server for evaluation. While a thin client might compose predefined fragments with user input filled in as parameters, a thick client (or another server) may allow more flexibility in the composition of requests.

There are several benefits associated with such an approach:

- The WCPS model has a clear, unambiguous semantics which is formally described; hence, interoperability risks are minimized as much as technically possible today.
- WCPS is coherent with WCS concepts; vendors may, for example, choose to offer only WCS or WCS plus additionally WCPS, all running on one and the same data asset.
- WCPS has a human-readable abstract syntax which incorporates the experience of SQL and XQuery. The abstract syntax translates into both XML and URL-encoded syntax; automatic translators have been implemented already.
- To accommodate the Use Cases described below, WCPS offers an extended coverage model: a WCPS coverage can have any number of dimensions, whereby non-spatiotemporal axes can be added by the application; this allows, among others, to integrate data warehousing/OLAP with the geo world.
- Customized profiles can be defined easily; technically, they consist of request fragments which are instantiated by a client by filling in parameters and/or further fragments. As an example, a WCPS can feed a WMS from WCS data, allowing for example dynamic addition and change of styles without human interference on server side.

- Automatic service dispatching, chaining, and optimizing become possible because the operation semantics is known to both client engines and servers. For example, a WCPS server is able to automatically extract portions that are best resolved locally, distribute other requested parts to other suitable servers, re-collect results, package them, and return them without any human interference.

WCPS is complementary to the WPS approach. While WPS is very general, WCPS focuses on coverages. WPS describes the interfaces (such as parameter types), but not the semantics of an operation in a machine-readable manner (the semantics of, say, an image overlay function is described verbally only, hence it can be evaluated by humans only); WCPS, on the contrary, adopts an XML structure which conveys operations and their chaining to the (non-human) agents involved.

In view of these characteristics, someone has termed WPS a “loosely coupled approach” and WCPS a “tightly coupled approach”. Notably both are well compatible; actually an annex of the WCPS document specifies the WPS wrapping of WCPS services.

## 6.2 Use Cases

The following use cases have been identified; as many as possible of them will be addressed within OWS-4:

- x/y slice from an x/y/z/t cube

  User addresses a 4-D x/y/z/t climate model and selects a slice along x and y axis at some random z and t position. The result is delivered as a TIFF image.

- x/z slice from an x/y/z/t cube

  User addresses a 4-D x/y/z/t climate model and selects a slice along x and z axis at some random y and t position. Notably the result has only one spatial axis (x, but no y). The result is delivered as a TIFF image.

- WCS functionality (to demonstrate coherence of WCS and WCPS)

  User sends a WCS request (containing a mix of the operations spatial/temporal subsetting, range subsetting, scaling, reprojection, and data format encoding) and expects a valid WCS response. By means of some hidden instance, however, the WCS request is translated into a WCPS request, evaluated, and returned to the user.

- Time series summary sequence from an x/y/z/t cube

  User addresses the air temperature of a 4-D x/y/z/t climate model and requests the average temperature for each time slice available. The result is a 1-D time series of scalar values.

- Deriving the NDVI from a Landsat scene

  The NDVI *N* over a Landsat scene *s* is given by the formula[1]

  $$N(s) = ( s.nir - s.red ) / ( s.nir + s.red )$$

  *N* has a range between –1 and +1. High values indicate vegetation; hence thresholding with a value close to the maximum highlights vegetation.

  User selects a Landsat scene and sends an NDVI request. Server responds with a binary coverage where 1 indicates vegetation and 0 no vegetation.

- Histogram

  User requests a histogram over one band of a Landsat scene. The histogram (see figure) contains 256 buckets and forms itself a 1-D coverage having only one (so-called *abstract*) axis with no spatiotemporal semantics. This histogram is to be returned as part of a WCPS coverage response.

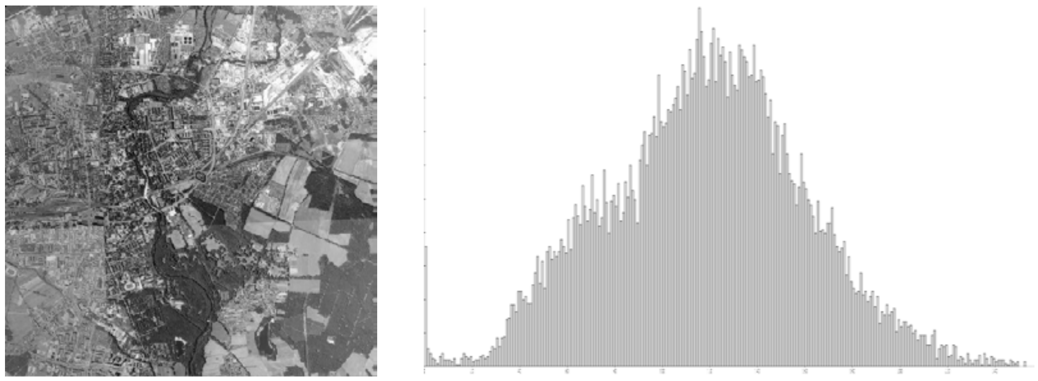

**Figure 12: Greyscale airborne image (left) and histogram (right)**

## 6.3 Design

IUB's WCPS implementation relies on standard (relational) database technology and the rasdaman raster server middleware.

The service has been implemented with core functionality being available on both client and server. The service stack (see figure below) makes use of standard Web components (Java; Tomcat; Xerces; JAXB), the rasdaman raster server middleware, and the PostgreSQL open-source database[2]. Notably all raster data are stored inside the PostgreSQL

[1] The dot notation ("*N.nir*") denotes the usual band extraction - in this case: the near infrared band.

[2] On principle any relational DBMS can serve this purpose, as rasdaman does not make use of any proprietary features; Oracle, Informix, DB2, and others have been used in practice.

database; rasdaman optimizes storage structures for fast retrieval and analysis and also performs dynamic query optimization.

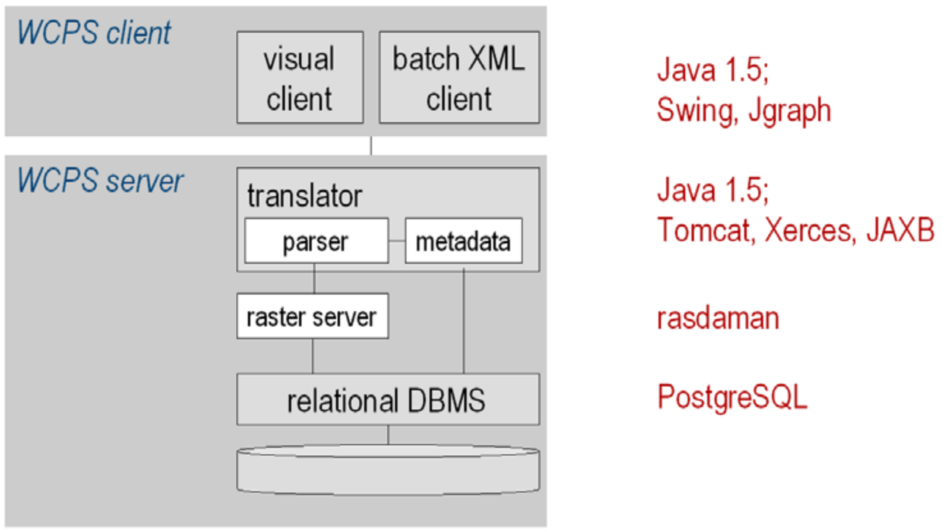

**Figure 13: WCPS service stack in the IUB implementation**

The server is structured as follows. A request is received via the translator component and is analysed, making use of some metadata (coverage names, dimensions, domains, ranges, etc.) stored in the relational database. After all syntax and semantic checks have been passed and the internal tree representation has been generated, the translator generates a rasql (rasdaman query language) request and sends this to the rasdaman server. Rasdaman optimizes this query, evaluates it making use of the coverage data sitting in the PostgreSQL database, and finally packages the result coverage. The translator component, finally, adds the WCPS metadata and ships back the response to the client.

This architecture opens up a wide space for internal optimization, as has been shown in a series of papers and PhD theses over the last five years.

To exploit the expressive power of WCPS it is not advisable to simply offer a point-and-click interface; users must be enabled to phrase complex requests too. On the other hand, graphical-interactive handling seems a must – a textual interface for typing in WCPS requests certainly is inadequate. It was decided, therefore, to develop a client based on Visual Programming concepts. This client uses Java technology (Swing, Jgraph) for graphically composing requests with automatic syntax checks. It is extensible through configuration files.

## 6.4 Results

Work in OWS-4 has led to a substantial step forward in both concepts and implementation. In the December 2006 OGC TC meeting progress has been reported and discussed including some of the lessons discussed in the next section.

The WCPS client and server are now available in its core functionality, with support for n-dimensional coverages. What is missing is support for coordinate systems other than ImageCRSs and specialized functionality like aggregates.

The current state can be demonstrated with the se case "on-the-fly derivation of the NDVI" (see Use Cases above). In WCPS Abstract Syntax an NDVI thresholding over Landsat data translates to the following request, assuming the result is to be delivered TIFF encoded:

```
for ls in ( Landsat_scene )
return
    encode( ( (ls.nir - l.red) / (ls.nir + ls.red) > 0.6 ), "TIFF" )
```

The following images show a false-coloured Landsat scene cut-out and the NDVI obtained by issuing the WCPS query described above:

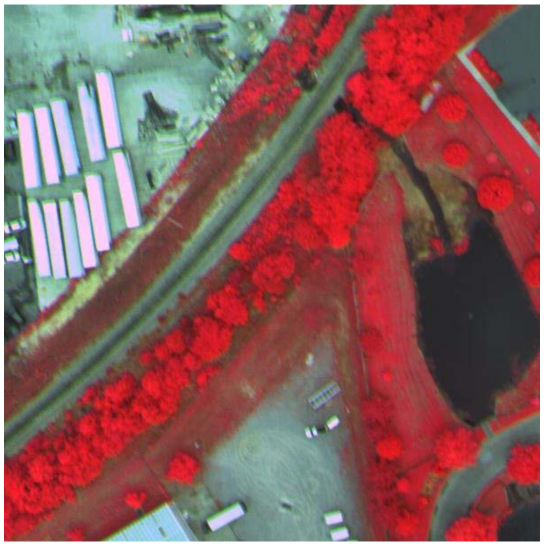

**Figure 14: False coloured Landsat image cut-out (obtained from screen shot)**

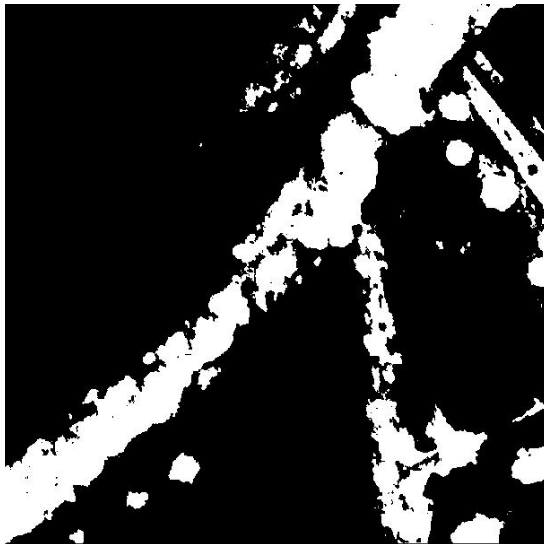

**Figure 15: NDVI-thresholded image (obtained from screen shot)**

## 6.5 Lessons Learned

While adapting WCPS to the new WCS 1.1, some observations have been made which will result in WCS and, where appropriate, OWS Common change requests:

- The coverage concept of OGC (and ISO) should allow coverages (i) with more than spatio-temporal axes and (ii) coverages with an arbitrary subset of spatio-temporal axes (such as x/z or y/t slices); this addresses OWS Common.

- The "null resistance" parameter in WCS 1.1 actually is closely tied to the interpolation type; therefore, both parameters should always occur pair wise. In WCS currently this is not the case for the default interpolation type.

- Similar to default interpolation, there should also be a default null value in WCS; currently a coverage may have a set of null values associated; in this case it is unclear which null value an operation (such as resampling/interpolation) actually should yield.

- CRSs frequently address only some of the coverage axes; e.g., a 4-D x/y/z/t coverage may be addressed via WGS84 (which cares only for x/y) and owsTime coordinates (which only cares about t); another CRS may address x/y/z – hence, any combination (i.e., any subset) of coverage axes basically can be the scope of a CRS. Hence, a more general modelling would tie CRSs to axes. Notably it makes sense that some axes always be addressed using the same CRS – x and y obviously are prime candidates for such a constraint.

  To this end, WCPS knows the concept of a supported CRS set *per axis*, mathematically: crsSet(C,a) for some coverage C and axis a. It is proposed to adopt this concept for WCS too.

- WCPS demands that a coverage always have an ImageCRS, regardless of addit-ional CRSs. This seems, for practical reasons, advantageous also for WCS.

Other additional requirements address WCPS itself:

- WCPS should allow for asynchronous processing.

## 6.6 Future Directions

Next steps include further work on the reference implementation, adding and implementing further use cases drawn from mapping, remote sensing, geophysics, and oceanography.

## 7 Build Context workflow

Early during the GPW work the group discussed a context building workflow. The idea was to create an OGC Context document that would be used refer to all of the data needed for certain use cases within the OWS-4 scenarios. It was never implemented.

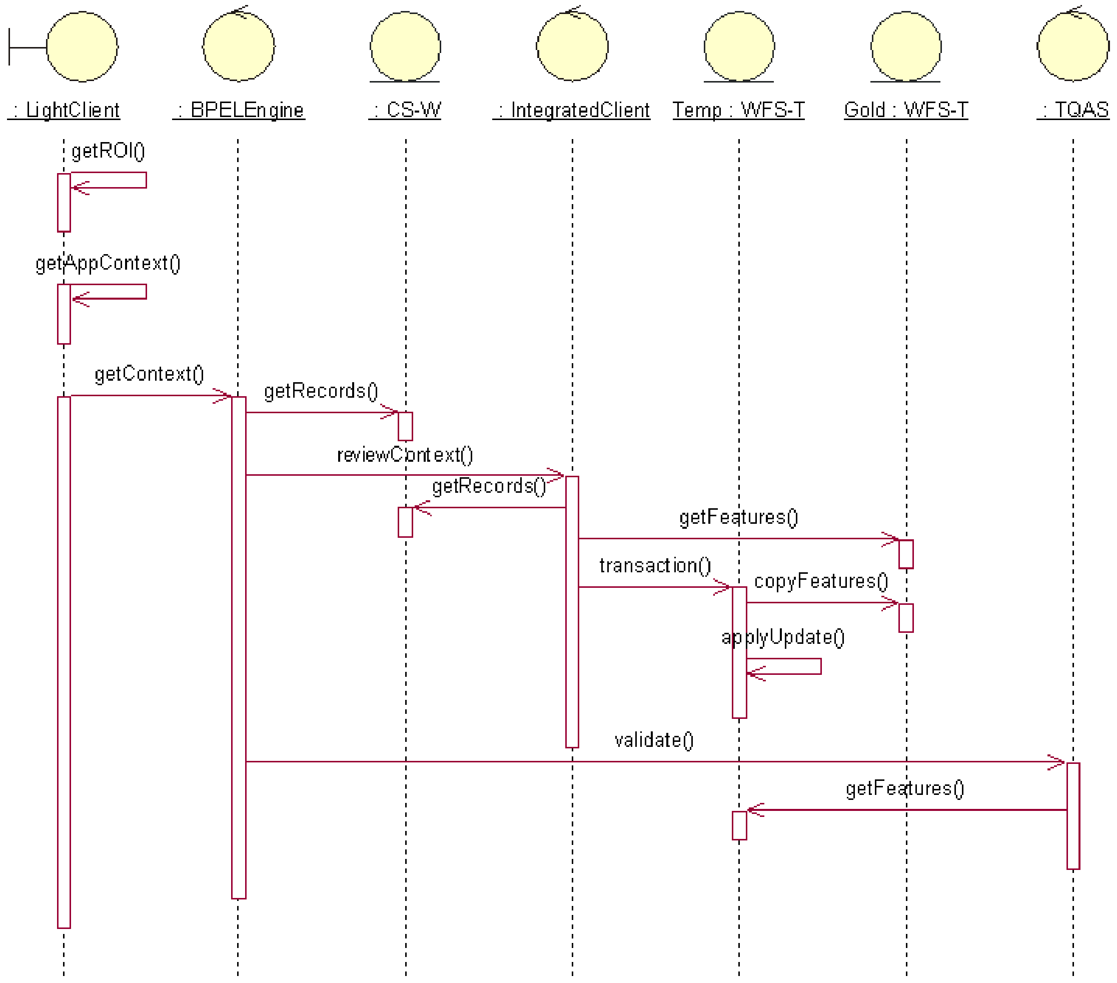

**Figure 16: Build context sequence diagram**

# 8 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:

### 8.1 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

### 8.2 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

### 8.3 adapter pattern

The adapter pattern adapts one interface for a class into one that a client expects. It is also called the wrapper pattern.

### 8.4 generalization

Generalization is the process adapting information on a map to the scale and display medium of the map.

# 9 Conventions

## 9.1 Symbols and abbreviations

| | |
|---|---|
| BPEL | Business Process Execution Language |
| GPW | Geo-Processing Workflow |
| KVP | Key Value Pair |
| OGC | Open Geospatial Consortium |
| OWS | OGC Web Services |
| OWS-4 | OGC Web Services interoperability program, phase 4 |
| NDVI | Normalized Difference Vegetation Index |
| REST | Representational State Transfer |
| SOAP | Simple Object Access Protocol |
| SSE | Service Support Environment |
| SWE | Sensor Web Enablement |
| TBC | To Be Confirmed |
| TBD | To Be Defined |
| WCS | Web Coverage Service |
| WCPS | Web Coverage Processing Service |
| WCTS | Web Coordinate Transformation Service |
| WNS | Web Notification Service |
| WPS | Web Processing Service |
| WSDL | Web Services Description Language |
| WSIF | Web Services Invocation Framework |
| XML | eXtensible Markup Language |

# Annex A WSDL

## A.1 GPW – WSDL for Data Reduction Workflow

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="clipping"
             targetNamespace="http://xmlns.oracle.com/clipping"
             xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:client="http://xmlns.oracle.com/clipping"
             xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/">

    <!-- -------------------------------------------------------------------
    TYPE DEFINITION - List of services participating in this BPEL process
    The default output of the BPEL designer uses strings as input and
    output to the BPEL Process.  But you can define or import any XML
    Schema type and us them as part of the message types.
    ------------------------------------------------------------------- -->
    <types>
        <schema attributeFormDefault="qualified"
        elementFormDefault="qualified"
        targetNamespace="http://xmlns.oracle.com/clipping"
        xmlns="http://www.w3.org/2001/XMLSchema">
        <element name="clippingProcessRequest">
         <complexType>
          <sequence>
           <element name="clippingMethod" type="string"/>
              <element name="dataURL" type="string"/>
              <element name="clippingURL" type="string"/>
          </sequence>
         </complexType>
        </element>
        <element name="clippingProcessResponse">
         <complexType>
          <sequence>
           <element name="result" type="string"/>
          </sequence>
         </complexType>
        </element>
        </schema>
    </types>

    <!-- -------------------------------------------------------------------
    MESSAGE TYPE DEFINITION - Definition of the message
    types used as part of the port type defintions
    ------------------------------------------------------------------- -->
    <message name="clippingRequestMessage">
        <part name="payload" element="client:clippingProcessRequest"/>
    </message>
    <message name="clippingResponseMessage">
        <part name="payload" element="client:clippingProcessResponse"/>
    </message>

    <!-- -------------------------------------------------------------------
    PORT TYPE DEFINITION - A port type groups a set of
    operations intoa logical service unit.
    ------------------------------------------------------------------- -->

    <!-- portType implemented by the clipping BPEL process -->
    <portType name="clipping">
        <operation name="process">
        <input  message="client:clippingRequestMessage" />
        <output message="client:clippingResponseMessage"/>
        </operation>
    </portType>

    <!-- -------------------------------------------------------------------
    PARTNER LINK TYPE DEFINITION
    ----------------------------------------------------------------------- -->
    <plnk:partnerLinkType name="clipping">
        <plnk:role name="clippingProvider">
        <plnk:portType name="client:clipping"/>
        </plnk:role>
    </plnk:partnerLinkType>
</definitions>
```

## A.2 GPW - WSDL for TQAS Workflow

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="TQAS"
             targetNamespace="http://xmlns.oracle.com/TQAS"
             xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:client="http://xmlns.oracle.com/TQAS"
             xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/">

    <!-- ------------------------------------------------------------------------
    TYPE DEFINITION - List of services participating in this BPEL process.
    The default output of the BPEL   designer uses strings as input and
    output to the BPEL Process.  But you can define or import any XML
    Schema type and us them as part of the message types.
    ----------------------------------------------------------------------- -->
    <types>
        <schema attributeFormDefault="qualified"
        elementFormDefault="qualified"
        targetNamespace="http://xmlns.oracle.com/TQAS"
        xmlns="http://www.w3.org/2001/XMLSchema">
        <element name="TQASProcessRequest">
         <complexType>
          <sequence>
           <element name="session_parent_id" type="string"/>
              <element name="data_store_ref_id" type="string"/>
              <element name="rule_ref_id" type="string"/>
              <element name="session_status" type="string"/>
          </sequence>
         </complexType>
        </element>
        <element name="TQASProcessResponse">
         <complexType>
          <sequence>
           <element name="result" type="string"/>
          </sequence>
         </complexType>
        </element>
        </schema>
    </types>

    <!-- ------------------------------------------------------------------------
    MESSAGE TYPE DEFINITION - Definition of the message
    types used as part of the port type defintions
    ----------------------------------------------------------------------- -->
    <message name="TQASRequestMessage">
        <part name="payload" element="client:TQASProcessRequest"/>
    </message>
    <message name="TQASResponseMessage">
        <part name="payload" element="client:TQASProcessResponse"/>
    </message>

    <!-- ------------------------------------------------------------------------
    PORT TYPE DEFINITION - A port type groups a set of
    operations into a logical service unit.
    ----------------------------------------------------------------------- -->

    <!-- portType implemented by the TQAS BPEL process -->
    <portType name="TQAS">
        <operation name="process">
        <input  message="client:TQASRequestMessage" />
        <output message="client:TQASResponseMessage"/>
        </operation>
    </portType>

    <!-- ---------------------------------------------------------------------
    PARTNER LINK TYPE DEFINITION
    --------------------------------------------------------------------- -->
    <plnk:partnerLinkType name="TQAS">
        <plnk:role name="TQASProvider">
         <plnk:portType name="client:TQAS"/>
        </plnk:role>
    </plnk:partnerLinkType>
</definitions>
```

## A.3 WSDL for EO1 Demo Workflow

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="EO1"
             targetNamespace="http://geobrain.laits.gmu.edu/EO1"
             xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:client="http://geobrain.laits.gmu.edu/EO1"
             xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/">

    <!-- ---------------------------------------------------------------------
    TYPE DEFINITION - List of services participating in this BPEL process
    The default output of the BPEL designer uses strings as input and
    output to the BPEL Process.  But you can define or import any XML
    Schema type and us them as part of the message types.
    --------------------------------------------------------------------- -->
    <types>
        <schema attributeFormDefault="qualified"
        elementFormDefault="qualified"
        targetNamespace="http://geobrain.laits.gmu.edu/EO1"
        xmlns="http://www.w3.org/2001/XMLSchema">
        <element name="EO1ProcessRequest">
         <complexType>
          <sequence>
           <element name="input" type="string"/>

          </sequence>
         </complexType>
        </element>
        <element name="EO1ProcessResponse">
         <complexType>
          <sequence>
           <element name="result" type="anyURI"/>
          </sequence>
         </complexType>
        </element>
        </schema>
    </types>

    <!-- ---------------------------------------------------------------------
    MESSAGE TYPE DEFINITION - Definition of the message types
    used as part of the port type definitions
    --------------------------------------------------------------------- -->
    <message name="EO1RequestMessage">
        <part name="payload" element="client:EO1ProcessRequest"/>
    </message>
    <message name="EO1ResponseMessage">
        <part name="payload" element="client:EO1ProcessResponse"/>
    </message>

    <!-- ---------------------------------------------------------------------
    PORT TYPE DEFINITION - A port type groups a set of
    operations into a logical service unit.
    --------------------------------------------------------------------- -->

    <!-- portType implemented by the EO1 BPEL process -->
    <portType name="EO1">
```

```
        <operation name="process">
        <input  message="client:EO1RequestMessage" />
        <output message="client:EO1ResponseMessage"/>
        </operation>
    </portType>

    <!-- -------------------------------------------------------------------
    PARTNER LINK TYPE DEFINITION
    ------------------------------------------------------------------- -->
    <plnk:partnerLinkType name="EO1">
        <plnk:role name="EO1Provider">
        <plnk:portType name="client:EO1"/>
        </plnk:role>
    </plnk:partnerLinkType>
</definitions>
```

# Annex B BPEL Scripts

## B.1 GPW – BPEL for Data Reduction Workflow

```
<process name="clipping" targetNamespace="http://xmlns.oracle.com/clipping"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.services.functions.Xpa
th20" xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:ns1="http://geobrain.laits.gmu.edu/schemas/wpsWSDL"
xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ns3="http://www.opengeospatial.net/ows"
xmlns:ns2="http://www.opengeospatial.net/wps"
xmlns:client="http://xmlns.oracle.com/clipping"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:ora="http://schemas.oracle.com/xpath/extension"
xmlns:orcl="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.services.functions.Ext
Func">
<!-- ================================================================= -->
<!-- PARTNERLINKS                                                      -->
<!-- List of services participating in this BPEL process               -->
<!-- ================================================================= -->
<partnerLinks>
    <partnerLink name="client" partnerLinkType="client:clipping"
myRole="clippingProvider"/>
    <partnerLink myRole="WPS_HTTP_Post_Port_Role" name="clipping_1"
partnerRole="WPS_HTTP_Post_Port_Role" partnerLinkType="ns1:WPS_HTTP_Post_Port_PL"/>
    <partnerLink myRole="WPS1_HTTP_Post_Port_Role" name="Generalization"
partnerRole="WPS1_HTTP_Post_Port_Role" partnerLinkType="ns1:WPS1_HTTP_Post_Port_PL"/>
  </partnerLinks>
<!-- ================================================================= -->
<!-- VARIABLES                                                         -->
<!-- List of messages and XML documents used within this BPEL process  -->
<!-- ================================================================= -->
  <variables>
    <!-- Reference to the message passed as input during initiation -->
    <variable name="inputVariable" messageType="client:clippingRequestMessage"/>
    <!--  Reference to the message that will be returned to the requester -->
    <variable name="outputVariable" messageType="client:clippingResponseMessage"/>
    <variable name="Invoke_Clipping_Execute_InputVariable"
messageType="ns1:ExecuteMessageIn"/>
    <variable name="Invoke_Clipping_Execute_OutputVariable"
messageType="ns1:ExecuteMessageOut"/>
    <variable name="Invoke_Generalization_Execute_InputVariable"
messageType="ns1:ExecuteMessageIn"/>
    <variable name="Invoke_Generalization_Execute_OutputVariable"
messageType="ns1:ExecuteMessageOut"/>
  </variables>
<!-- ================================================================= -->
<!-- ORCHESTRATION LOGIC                                               -->
<!-- Set of activities coordinating the flow of messages across the    -->
<!-- services integrated within this business process                  -->
<!-- ================================================================= -->
  <sequence name="main">
    <!-- Receive input from requestor.
    Note: This maps to operation defined in clipping.wsdl
    -->
    <receive name="receiveInput" partnerLink="client" portType="client:clipping"
operation="process" variable="inputVariable" createInstance="yes"/>
    <!-- Generate reply to synchronous request -->
    <assign name="Assign_2">
      <copy>
        <from expression="'WPS'"/>
        <to variable="Invoke_Generalization_Execute_InputVariable" part="parameters"
query="/ns2:Execute/@service"/>
      </copy>
      <copy>
        <from expression="'false'"/>
        <to variable="Invoke_Generalization_Execute_InputVariable" part="parameters"
query="/ns2:Execute/@status"/>
      </copy>
      <copy>
        <from expression="'0.4.0'"/>
        <to variable="Invoke_Generalization_Execute_InputVariable" part="parameters"
query="/ns2:Execute/@version"/>
      </copy>
      <copy>
        <from
expression="'org.n52.wps.server.algorithm.simplify.DouglasPeuckerAlgorithm'"/>
        <to variable="Invoke_Generalization_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns3:Identifier"/>
      </copy>
      <copy>
        <from expression="'true'"/>
        <to variable="Invoke_Generalization_Execute_InputVariable" part="parameters"
query="/ns2:Execute/@store"/>
      </copy>
      <copy>
        <from expression="'FEATURES'"/>
        <to variable="Invoke_Generalization_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:DataInputs/ns2:Input[1]/ns3:Identifier"/>
      </copy>
      <copy>
        <from expression="'asd'"/>
        <to variable="Invoke_Generalization_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:DataInputs/ns2:Input[1]/ns3:Title"/>
      </copy>
      <copy>
        <from variable="inputVariable" part="payload"
query="/client:clippingProcessRequest/client:dataURL"/>
        <to variable="Invoke_Generalization_Execute_InputVariable"
query="/ns2:Execute/ns2:DataInputs/ns2:Input[1]/ns2:ComplexValueReference/@ns3:reference"
/>
      </copy>
      <copy>
        <from expression="'http://schemas.opengis.net/gml/2.1.2/feature.xsd'"/>
        <to variable="Invoke_Generalization_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:DataInputs/ns2:Input[1]/ns2:ComplexValueReference/@schema"/>
      </copy>
      <copy>
        <from expression="'TOLERANCE'"/>
        <to variable="Invoke_Generalization_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:DataInputs/ns2:Input[2]/ns3:Identifier"/>
      </copy>
      <copy>
        <from expression="'asd'"/>
        <to variable="Invoke_Generalization_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:DataInputs/ns2:Input[2]/ns3:Title"/>
      </copy>
      <copy>
        <from expression="'meters'"/>
        <to variable="Invoke_Generalization_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:DataInputs/ns2:Input[2]/ns2:LiteralValue/@uom"/>
      </copy>
      <copy>
        <from expression="'xs:double'"/>
        <to variable="Invoke_Generalization_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:DataInputs/ns2:Input[2]/ns2:LiteralValue/@dataType"/>
      </copy>
      <copy>
        <from expression="'0.000062'"/>
        <to variable="Invoke_Generalization_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:DataInputs/ns2:Input[2]/ns2:LiteralValue"/>
      </copy>
      <copy>
        <from expression="'http://schemas.opengis.net/gml/2.1.2/feature.xsd'"/>
        <to variable="Invoke_Generalization_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:OutputDefinitions/ns2:Output/@schema"/>
      </copy>
      <copy>
        <from expression="'SIMPLIFIED_FEATURES'"/>
        <to variable="Invoke_Generalization_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:OutputDefinitions/ns2:Output/ns3:Identifier"/>
      </copy>
    </assign>
    <invoke name="Invoke_Generalization" partnerLink="Generalization"
portType="ns1:WPS1_HTTP_Post_Port" operation="Execute"
inputVariable="Invoke_Generalization_Execute_InputVariable"
outputVariable="Invoke_Generalization_Execute_OutputVariable"/>
    <assign name="Assign_1">
      <copy>
        <from expression="'WPS'"/>
        <to variable="Invoke_Clipping_Execute_InputVariable" part="parameters"
query="/ns2:Execute/@service"/>
      </copy>
      <copy>
        <from expression="'true'"/>
        <to variable="Invoke_Clipping_Execute_InputVariable" part="parameters"
query="/ns2:Execute/@store"/>
      </copy>
      <copy>
        <from expression="'false'"/>
        <to variable="Invoke_Clipping_Execute_InputVariable" part="parameters"
query="/ns2:Execute/@status"/>
      </copy>
      <copy>
        <from expression="'0.4.0'"/>
        <to variable="Invoke_Clipping_Execute_InputVariable" part="parameters"
query="/ns2:Execute/@version"/>
      </copy>
      <copy>
        <from variable="inputVariable" part="payload"
query="/client:clippingProcessRequest/client:clippingMethod"/>
        <to variable="Invoke_Clipping_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns3:Identifier"/>
      </copy>
      <copy>
        <from expression="'aInputvector'"/>
        <to variable="Invoke_Clipping_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:DataInputs/ns2:Input[1]/ns3:Identifier"/>
      </copy>
      <copy>
        <from expression="'Input vector map'"/>
        <to variable="Invoke_Clipping_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:DataInputs/ns2:Input[1]/ns3:Title"/>
      </copy>
      <copy>
        <from expression="'xs:string'"/>
        <to variable="Invoke_Clipping_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:DataInputs/ns2:Input[1]/ns2:LiteralValue/@dataType"/>
      </copy>
      <copy>
        <from variable="Invoke_Clipping_Execute_OutputVariable" part="parameters"
query="/ns2:ExecuteResponse/ns2:ProcessOutputs/ns2:Output/ns2:ComplexValueReference/@ns3:
reference"/>
        <to variable="Invoke_Clipping_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:DataInputs/ns2:Input[1]/ns2:LiteralValue"/>
      </copy>
      <copy>
        <from expression="'bInputvector'"/>
        <to variable="Invoke_Clipping_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:DataInputs/ns2:Input[2]/ns3:Identifier"/>
      </copy>
      <copy>
        <from expression="'Input vector map'"/>
        <to variable="Invoke_Clipping_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:DataInputs/ns2:Input[2]/ns3:Title"/>
      </copy>
      <copy>
        <from expression="'xs:string'"/>
        <to variable="Invoke_Clipping_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:DataInputs/ns2:Input[2]/ns2:LiteralValue/@dataType"/>
      </copy>
      <copy>
        <from variable="inputVariable" part="payload"
query="/client:clippingProcessRequest/client:clippingURL"/>
        <to variable="Invoke_Clipping_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:DataInputs/ns2:Input[2]/ns2:LiteralValue"/>
      </copy>
      <copy>
        <from expression="'ClipOperator'"/>
        <to variable="Invoke_Clipping_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:DataInputs/ns2:Input[3]/ns3:Identifier"/>
      </copy>
      <copy>
        <from expression="'Grass v.overlay command operator'"/>
        <to variable="Invoke_Clipping_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:DataInputs/ns2:Input[3]/ns3:Title"/>
      </copy>
      <copy>
        <from expression="'xs:string'"/>
        <to variable="Invoke_Clipping_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:DataInputs/ns2:Input[3]/ns2:LiteralValue/@dataType"/>
      </copy>
      <copy>
        <from expression="'And'"/>
        <to variable="Invoke_Clipping_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:DataInputs/ns2:Input[3]/ns2:LiteralValue"/>
      </copy>
      <copy>
        <from expression="'ClipResult'"/>
        <to variable="Invoke_Clipping_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:OutputDefinitions/ns2:Output/ns3:Identifier"/>
      </copy>
      <copy>
        <from expression="'Test Output'"/>
        <to variable="Invoke_Clipping_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:OutputDefinitions/ns2:Output/ns3:Title"/>
      </copy>
      <copy>
        <from expression="'URL to a output vector file'"/>
        <to variable="Invoke_Clipping_Execute_InputVariable" part="parameters"
query="/ns2:Execute/ns2:OutputDefinitions/ns2:Output/ns3:Abstract"/>
      </copy>
    </assign>
    <invoke name="Invoke_clipping" partnerLink="clipping_1"
portType="ns1:WPS_HTTP_Post_Port" operation="Execute"
inputVariable="Invoke_Clipping_Execute_InputVariable"
outputVariable="Invoke_Clipping_Execute_OutputVariable"/>
    <assign name="Assign_3">
      <copy>
        <from variable="outputVariable" part="payload"
query="/client:clippingProcessResponse/client:result"/>
        <to variable="Invoke_Clipping_Execute_OutputVariable" part="parameters"
query="/ns2:ExecuteResponse/ns2:ProcessOutputs/ns2:Output/ns2:LiteralValue"/>
      </copy>
    </assign>
    <reply name="replyOutput" partnerLink="client" portType="client:clipping"
operation="process" variable="outputVariable"/>
  </sequence>
</process>
```

## B.2 GPW – BPEL for TQAS Workflow

```
<process name="TQAS" targetNamespace="http://xmlns.oracle.com/TQAS"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.services.functions.Xpa
th20" xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:ns1="http://laserscan.com/RadiusStudio"
xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:client="http://xmlns.oracle.com/TQAS"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:ora="http://schemas.oracle.com/xpath/extension"
xmlns:orcl="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.services.functions.Ext
Func">
<!-- ================================================================= -->
<!-- PARTNERLINKS                                                      -->
<!-- List of services participating in this BPEL process               -->
<!-- ================================================================= -->
  <partnerLinks>
    <!--
    The 'client' role represents the requester of this service.  It is
    used for callback.  The location and correlation information associated
    with the client role are automatically set using WS-Addressing.
    -->
    <partnerLink name="client" partnerLinkType="client:TQAS" myRole="TQASProvider"/>
    <partnerLink myRole="SessionManagerEndpoint_Role" name="TQAS"
partnerRole="SessionManagerEndpoint_Role"
partnerLinkType="ns1:SessionManagerEndpoint_PL"/>
  </partnerLinks>
<!-- ================================================================= -->
<!-- VARIABLES                                                         -->
<!-- List of messages and XML documents used within this BPEL process  -->
<!-- ================================================================= -->
  <variables>
    <!-- Reference to the message passed as input during initiation -->
    <variable name="inputVariable" messageType="client:TQASRequestMessage"/>
    <!-- Reference to the message that will be returned to the requester -->
    <variable name="outputVariable" messageType="client:TQASResponseMessage"/>
    <variable name="Create_Session_create_InputVariable"
messageType="ns1:SessionManagerEndpoint_create"/>
    <variable name="Create_Session_create_OutputVariable"
messageType="ns1:SessionManagerEndpoint_createResponse"/>
    <variable name="Run_Session_run_InputVariable"
messageType="ns1:SessionManagerEndpoint_run"/>
    <variable name="Run_Session_run_OutputVariable"
messageType="ns1:SessionManagerEndpoint_runResponse"/>
    <variable name="Monitor_Session_getProgress_InputVariable"
messageType="ns1:SessionManagerEndpoint_getProgress"/>
    <variable name="Monitor_Session_getProgress_OutputVariable"
messageType="ns1:SessionManagerEndpoint_getProgressResponse"/>
    <variable name="Check_Session_Results_getResults_InputVariable"
messageType="ns1:SessionManagerEndpoint_getResults"/>
    <variable name="Check_Session_Results_getResults_OutputVariable"
messageType="ns1:SessionManagerEndpoint_getResultsResponse"/>
  </variables>
<!-- ================================================================= -->
<!-- ORCHESTRATION LOGIC                                               -->
<!-- Set of activities coordinating the flow of messages across the    -->
<!-- services integrated within this business process                  -->
<!-- ================================================================= -->
  <sequence name="main">
    <!-- Receive input from requestor.
    Note: This maps to operation defined in TQAS.wsdl
    -->
    <receive name="receiveInput" partnerLink="client" portType="client:TQAS"
operation="process" variable="inputVariable" createInstance="yes"/>
    <!-- Generate reply to synchronous request -->
    <assign name="Assign_Create_Session">
      <copy>
```

```
        <from expression="'&lt;![CDATA[&lt;Metadata&gt;&lt;Name value=&quot;Test
Session&quot;/&gt;&lt;Description value=&quot;Created by TQAS Web
Service&quot;/&gt;&lt;Comments value=&quot;Should be deleted after
use.&quot;/&gt;&lt;/Metadata&gt;]]&gt;'"/>
        <to variable="Create_Session_create_InputVariable" part="DTO_1"
query="/DTO_1/metadataXml"/>
      </copy>
      <copy>
        <from expression="'Test Session'"/>
        <to variable="Create_Session_create_InputVariable" part="DTO_1"
query="/DTO_1/name"/>
      </copy>
      <copy>
        <from variable="inputVariable" part="payload"
query="/client:TQASProcessRequest/client:session_parent_id"/>
        <to variable="Create_Session_create_InputVariable" part="DTO_1"
query="/DTO_1/parentId"/>
      </copy>
      <copy>
        <from expression="'1'"/>
        <to variable="Create_Session_create_InputVariable" part="DTO_1"
query="/DTO_1/sequence"/>
      </copy>
      <copy>
        <from expression="'&lt;![CDATA[&lt;Session&gt;&lt;Sequence&gt;&lt;Task
label=&quot;1&quot; type=&quot;Open Data&quot;&gt;&lt;DataStoreRef
ref_id=&quot;0ac3a94ac0a85abd0181f53b7fc2e033&quot;/&gt;&lt;/Task&gt;&lt;Task
label=&quot;2&quot; type=&quot;Open Data&quot;&gt;&lt;DataStoreRef
ref_id=&quot;0ac98ebdc0a85abd019732afc38ace6e&quot;/&gt;&lt;/Task&gt;&lt;Task
label=&quot;3&quot; type=&quot;Check Rules&quot;&gt;&lt;RuleRef
ref_id=&quot;1a714a8ec0a85abd01dafa55d327e7be&quot;/&gt;&lt;/Task&gt;&lt;/Sequence&gt;&lt
;/Session&gt;]]&gt;'"/>
        <to variable="Create_Session_create_InputVariable" part="DTO_1"
query="/DTO_1/xml"/>
      </copy>
    </assign>
    <invoke name="Create_Session" partnerLink="TQAS"
portType="ns1:SessionManagerEndpoint" operation="create"
inputVariable="Create_Session_create_InputVariable"
outputVariable="Create_Session_create_OutputVariable"/>
    <assign name="Assign_Run_Session">
      <copy>
        <from variable="Create_Session_create_OutputVariable" part="result"
query="/result/id"/>
        <to variable="Run_Session_run_InputVariable" part="String_1"/>
      </copy>
    </assign>
    <invoke name="Run_Session" partnerLink="TQAS" portType="ns1:SessionManagerEndpoint"
operation="run" inputVariable="Run_Session_run_InputVariable"
outputVariable="Run_Session_run_OutputVariable"/>
    <while name="While_1"
condition="bpws:getVariableData('inputVariable','payload','/client:TQASProcessRequest/cli
ent:session_status')contains('Finished')">
      <sequence name="Sequence_1">
        <wait name="Wait_Run_Session" for="'PT15S'"/>
        <assign name="Assign_Monitor_Session">
          <copy>
            <from variable="Create_Session_create_OutputVariable" part="result"
query="/result/id"/>
            <to variable="Monitor_Session_getProgress_InputVariable" part="String_1"/>
          </copy>
        </assign>
        <invoke name="Monitor_Session" partnerLink="TQAS"
portType="ns1:SessionManagerEndpoint" operation="getProgress"
inputVariable="Monitor_Session_getProgress_InputVariable"
outputVariable="Monitor_Session_getProgress_OutputVariable"/>
        <assign name="Assign_Session_Status">
          <copy>
            <from variable="Monitor_Session_getProgress_OutputVariable" part="result"/>
            <to variable="inputVariable" part="payload"
query="/client:TQASProcessRequest/client:session_status"/>
```

```
          </copy>
        </assign>
      </sequence>
    </while>
    <assign name="Assign_Check_Session">
      <copy>
        <from variable="Create_Session_create_OutputVariable" part="result"
query="/result/id"/>
        <to variable="Check_Session_Results_getResults_InputVariable" part="String_1"/>
      </copy>
      <copy>
        <from expression="'3'"/>
        <to variable="Check_Session_Results_getResults_InputVariable" part="String_2"/>
      </copy>
      <copy>
        <from expression="'1'"/>
        <to variable="Check_Session_Results_getResults_InputVariable" part="int_3"/>
      </copy>
      <copy>
        <from expression="'0'"/>
        <to variable="Check_Session_Results_getResults_InputVariable" part="int_4"/>
      </copy>
    </assign>
    <invoke name="Check_Session_Results" partnerLink="TQAS"
portType="ns1:SessionManagerEndpoint" operation="getResults"
inputVariable="Check_Session_Results_getResults_InputVariable"
outputVariable="Check_Session_Results_getResults_OutputVariable"/>
    <assign name="Assign_result">
      <copy>
        <from variable="Check_Session_Results_getResults_OutputVariable" part="result"/>
        <to variable="outputVariable" part="payload"
query="/client:TQASProcessResponse/client:result"/>
      </copy>
    </assign>
    <reply name="replyOutput" partnerLink="client" portType="client:TQAS"
operation="process" variable="outputVariable"/>
  </sequence>
</process>
```

## B.3 BPEL for EO1 Demo Worflow

```
<process name="EO1" targetNamespace="http://geobrain.laits.gmu.edu/EO1"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.services.functions.Xpa
th20" xmlns:ns4="http://datafed.net/xs/MapGridOperator"
xmlns:ns7="http://datafed.net/xs/Lineage"
xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:ns5="http://www.opengis.net/wcts"
xmlns:client="http://geobrain.laits.gmu.edu/EO1" xmlns:ns6="http://datafed.net/xs/Cube"
xmlns:ora="http://schemas.oracle.com/xpath/extension"
xmlns:ns1="http://geobrain.laits.gmu.edu/schemas/wpsWSDL"
xmlns:ns3="http://www.opengeospatial.net/ows"
xmlns:ns2="http://www.opengeospatial.net/wps"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:orcl="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.services.functions.Ext
Func">
<!-- ================================================================ -->
<!-- PARTNERLINKS                                                      -->
<!-- List of services participating in this BPEL process               -->
<!-- ================================================================ -->
  <partnerLinks>
    <!--
    The 'client' role represents the requester of this service.  It is
    used for callback.  The location and correlation information associated
    with the client role are automatically set using WS-Addressing.
    -->
    <partnerLink name="client" partnerLinkType="client:EO1" myRole="EO1Provider"/>
    <partnerLink myRole="MapGridOperatorSoap_Role" name="GridOperation"
partnerRole="MapGridOperatorSoap_Role" partnerLinkType="ns4:MapGridOperatorSoap_PL"/>
  </partnerLinks>
<!-- ================================================================= -->
<!-- VARIABLES                                                         -->
<!-- List of messages and XML documents used within this BPEL process  -->
<!-- ================================================================= -->
  <variables>
    <!-- Reference to the message passed as input during initiation -->
    <variable name="inputVariable" messageType="client:EO1RequestMessage"/>
    <!--
    Reference to the message that will be returned to the requester
    -->
    <variable name="outputVariable" messageType="client:EO1ResponseMessage"/>
    <variable name="Invoke_GridOperation_Evaluate_InputVariable"
messageType="ns4:EvaluateSoapIn"/>
    <variable name="Invoke_GridOperation_Evaluate_OutputVariable"
messageType="ns4:SoapOut"/>
  </variables>
<!-- ================================================================= -->
<!-- ORCHESTRATION LOGIC                                               -->
<!-- Set of activities coordinating the flow of messages across the    -->
<!-- services integrated within this business process                  -->
<!-- ================================================================= -->
  <sequence name="main">
    <!-- Receive input from requestor.
    Note: This maps to operation defined in EO1.wsdl
    -->
    <receive name="receiveInput" partnerLink="client" portType="client:EO1"
operation="process" variable="inputVariable" createInstance="yes"/>
    <!-- Generate reply to synchronous request -->
    <assign name="Assign_GridOperation">
      <copy>
        <from expression="'a/b'"/>
        <to variable="Invoke_GridOperation_Evaluate_InputVariable" part="parameters"
query="/ns4:Evaluate/ns4:expression"/>
      </copy>
      <copy>
        <from
expression="'http://webapps.datafed.net/ogc_NASA.wsfl?SERVICE=WCS&amp;amp;REQUEST=GetCove
rage&amp;amp;VERSION=1.0.0&amp;amp;CRS=EPSG:4326&amp;amp;COVERAGE=Hyperion_EO1.B021&amp;a
mp;TIME=2004-03-26T00:00:00&amp;amp;WIDTH=-1&amp;amp;HEIGHT=-1&amp;amp;DEPTH=-
1&amp;amp;FORMAT=NetCDF'"/>
        <to variable="Invoke_GridOperation_Evaluate_InputVariable" part="parameters"
query="/ns4:Evaluate/ns4:grid0/ns6:CubeRef"/>
      </copy>
      <copy>
        <from
expression="'http://webapps.datafed.net/ogc_NASA.wsfl?SERVICE=WCS&amp;amp;REQUEST=GetCove
rage&amp;amp;VERSION=1.0.0&amp;amp;CRS=EPSG:4326&amp;amp;COVERAGE=Hyperion_EO1.B051&amp;a
mp;TIME=2004-03-26T00:00:00&amp;amp;WIDTH=-1&amp;amp;HEIGHT=-1&amp;amp;DEPTH=-
1&amp;amp;FORMAT=NetCDF'"/>
        <to variable="Invoke_GridOperation_Evaluate_InputVariable" part="parameters"
query="/ns4:Evaluate/ns4:grid1/ns6:CubeRef"/>
      </copy>
      <copy>
        <from expression="'lat lon'"/>
        <to variable="Invoke_GridOperation_Evaluate_InputVariable" part="parameters"
query="/ns4:Evaluate/ns4:GridAdjustSettings/ns4:Dimensions"/>
      </copy>
      <copy>
        <from expression="'must_match'"/>
        <to variable="Invoke_GridOperation_Evaluate_InputVariable" part="parameters"
query="/ns4:Evaluate/ns4:GridAdjustSettings/ns4:ZoomStyle"/>
      </copy>
      <copy>
        <from expression="''"/>
        <to variable="Invoke_GridOperation_Evaluate_InputVariable" part="parameters"
query="/ns4:Evaluate/ns4:GridAdjustSettings/ns4:Zoom"/>
      </copy>
      <copy>
```

```
        <from expression="'use_default'"/>
        <to variable="Invoke_GridOperation_Evaluate_InputVariable" part="parameters"
query="/ns4:Evaluate/ns4:GridAdjustSettings/ns4:SizeStyle"/>
      </copy>
      <copy>
        <from expression="''"/>
        <to variable="Invoke_GridOperation_Evaluate_InputVariable" part="parameters"
query="/ns4:Evaluate/ns4:GridAdjustSettings/ns4:Size"/>
      </copy>
      <copy>
        <from expression="'GeoTIFF'"/>
        <to variable="Invoke_GridOperation_Evaluate_InputVariable" part="parameters"
query="/ns4:Evaluate/ns4:format"/>
      </copy>
    </assign>
    <invoke name="Invoke_GridOperation" partnerLink="GridOperation"
portType="ns4:MapGridOperatorSoap" operation="Evaluate"
inputVariable="Invoke_GridOperation_Evaluate_InputVariable"
outputVariable="Invoke_GridOperation_Evaluate_OutputVariable"/>
    <assign name="Assign_Output">
      <copy>
        <from variable="Invoke_GridOperation_Evaluate_OutputVariable" part="parameters"
query="/ns4:Response/ns6:Cube/ns6:CubeRef"/>
        <to variable="outputVariable" part="payload"
query="/client:EO1ProcessResponse/client:result"/>
      </copy>
    </assign>
    <reply name="replyOutput" partnerLink="client" portType="client:EO1"
operation="process" variable="outputVariable"/>
  </sequence>
</process>
```

## Bibliography

[1] OWS Common Implementation Specification [OGC 05-008c1], 2005-11-22, https://portal.opengeospatial.org/files/?artifact_id=8798

[2] Topic 12 - The OpenGIS Service Architecture [OGC 02-112], 2001-09-14, http://portal.opengeospatial.org/files/?artifact_id=1221

[3] OWS2 IH4DS Service Chaining IPR [OGC O4-078], 2004-10-05, David S. Rosinger

[4] BPEL for Image Handling IPR, 2004-05-25, John Vincent

[5] OWS-3 Imagery Workflow Discussion Paper [05-140r1], 2006-04-21, Yves Coene

[6] OWS-3 Imagery Workflow IPR [05-140], 2006-03-30, Yves Coene

[7] Web Processing Service (WPS) [OGC 05-007r4], 2005-12-23, Peter Schut, Arliss Whiteside

[8] OWS-4 WPS IPR [OGC 06-182r1], 2007-01-05, Steven Keens, http://portal.opengeospatial.org/files/?artifact_id=19157&version=1

[9] OWS-4 Sensor Planning Service for EO: Enhanced Service Infrastructure Technology Architecture and Standards in the OWS-4 Test bed [OGC 07-008], January 2007, The Hoa Nguyen, Philippe Merigot, Marc Gilles, http://portal.opengeospatial.org/files/?artifact_id=19756&version=1

[10] Web Coverage Processing Service (WCPS) 0.0.3 [OGC 06-035r1], 2006-07-26, Peter Baumann, http://portal.opengeospatial.org/files/?artifact_id=14895